HARDWARE SOLUTIONS FOR NEXT GENERATION OF TELECOMMUNICATION SYSTEMS PHYSICAL LAYER IMPLEMENTATION

A Dissertation

by

EHSAN ROHANI

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Gwan S. Choi
Co-Chair of Committee,	Mi Lu
Committee Members,	Krishna Narayanan
	Duncan M. (Hank) Walker
Head of Department,	Miroslav M. Begovic

December 2017

Major Subject: Electrical and Computer Engineering

ABSTRACT

In telecommunication systems the goal is to increase the throughput of data communication. Multiple input and multiple output antennas (MIMO) can be used to increase the SNR of the system and as the result increase the throughput via increasing the quality of the service (QoS) or it can be used to increase the speed of transmitted data and the throughput as the result. This trade-off can be used as advantage when dealing with a system design to better utilize hardware/software resources as well as to optimally exploit the environmental factors associated with communication medium.

We studied the iterative detection and decoding of the received signals. In this case the received signals are being detected and decoded multiple times which results in higher complexity, and QoS. Considering the original complexity of the MIMO detectors hardware implementation of this method is challenging.

To reduce the complexity while expecting performance gain over traditional receivers like Minimum Mean Square of Errors (MMSE) and Zero Forcing (ZF) detectors we had to look at more recent detectors and accept some performance loss. These are detectors with close to ML performance like Kbest or amended versions or linear detectors like the LR aided (lattice reduction) MMSE, or LORD (layered orthogonal lattice detector) detector. Following this approach we considered Integer Forcing (IF) detectors.

The IF detectors are categorized as linear detectors; however, there is a search involved in calculating and generating the inverse of the channel matrix that searches for inverse of any full rank integer combination of transmitted data with minimum effect on channel

noise. We introduce an algorithm that is able to provide LLRs (soft values) for the detected signal. We show that these type of detectors can reduce the gap between Zero Forcing (ZF) linear detector and maximum likelihood (ML) detector to 48% with only 2.96$\times$ more complexity while ML detector is 28.86$\times$ more complex.

We have implemented an IF processor to evaluate the hardware performance of this detector. To accomplish this task we have improved the existing designs for singular value decomposer (SVD) and advanced the decomposer to achieve a high performance IF detector. We finally present a proof of consent for asynchronous implementation of MIMO satellite communication.

DEDICATION

To my wife Mona.

ACKNOWLEDGMENTS

Over the past seven years I have received support and encouragement from a great number of individuals. My co-advisors Professor Choi and Professor Lu who guided me through all the steps. Professor Choi has been a mentor, colleague, and friend and his guidance has made this a thoughtful and rewarding journey. Professor Lu came to my help when things got hard or I was facing a problem I could not solve on my own. I do appreciate her continuous support of my Ph.D studies and related research.

I would like to thank my dissertation committee members of Dr. Narayanan and Dr. Walker for their support as I moved from an idea to a completed study and for their insightful comments and encouragement. I would like to express my sincere gratitude to Professor Narayanan for his suggestion for looking to Integer-Forcing detectors and his insights, guides, and helps through the project. It is only fair if I thank Professor Duncan M. (Hank) Walker, while I had the course Testing and Diagnosis of Digital Systems he suggested that I look to approximate arithmetic and error tolerable implementation and this formed the base of my knowledge on this area which facilitate the contributions on Singular Value Decomposition.

I also, like to thank Professor Peng Li, he provided me with professional tips during our collaboration with his group on SVM project. In addition, I like to appreciate Professor Gratz, and Sprintson for their mentorship during the time I was instructor of the record for Computer Architecture and Design course.

I would like to express the deepest appreciation to Professor Khatri, my mentor in the

Academy of Future Faculties program, he helped and guided me in taking next steps in developing my future career.

Yu-Chih Huang, and Nihat Engin Tunali provided needed encouragement and insights and it was their help which gave me a good understanding of the Integer-Forcing detectors and also the necessary theoretical background. I thank my fellow lab mates Yoon Seok Yang, Jingwei Xu, and Tiben Che for the stimulating discussions, for the sleepless nights we were working together, and for all the fun we have had in the last couple of years.

Last but not the least, I would like to thank my family, my parents for supporting me spiritually throughout writing this thesis and my wife for all her patience during these years, especially since the birth of our daughter and finally my little angel who brought me ultimate joy and happiness.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Professor Gwan S. Choi, Mi Lu, and Krishna Narayanan of the Department of Electrical and Computer Engineering and Professor Duncan M. (Hank) Walker of the Department of Department of Computer Science and Engineering.

The Integer Forcing detectors was first introduced to us by Professor Narayanan. Jingwei Xu, and Tiben Che were very helpful in preparation of material in section 2 and 6.

All other work conducted for the thesis (or) dissertation was completed by the student independently.

Funding Sources

Financial and technical support for the patenting process of the material presented in section 4 is provided by the Office of Technology Commercialization at Texas A&M University.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION AND LITERATURE REVIEW

In this work we look at some of the computationally intensive algorithms that seems to be holding the solution for the next-generation of telecommunication systems. In all the cases using multiple antennas in transmitter and receiver multiple input multiple output(MIMO) is the center of the focus. Employing MIMO technique comes with the price of increased complexity. This increased complexity is not only due to the increase in data transfer but also is due to the complex algorithms that we are using for detecting and decoding the data. In general the complexity of the system increases exponentially with increase in data rates.

Multiple antennas can be utilized to increase the accuracy of the data received and as a result, increase the data throughput or it can be used as a means to increase the data transfer speed. There are different algorithms that are introduced for detecting and decoding the data while using multiple antennas. Some of these algorithms are more computationally heavy and as the result provide a better performance curve and the others are less demanding but they have compromised performance curves.

We have looked at both solutions with compare and contrast option and implemented some of the most beneficiary of them. We also, looked at some of the practical problems that might happen in the telecommunication systems and have provided proper solutions to them.

As the proof-of-concept we looked at the asynchronous implementation of some designs that particularly might be beneficial from asynchronous implementation. The large scale industrial implementation of asynchronous circuits using computer tools is not a common practice. Largely due to the lack of computer aided design tools; however, with this works we show how much some application might benefit from using such a technique in their implementation.

1.1 Motivation

With the advancement in the use of telecommunication systems and the vast and increasing use of the internet, optimum implementation of the hardware that is used as the back bone of this advancement is really important. The optimum implementation of the hardware can promote the use of these systems and internet with decreasing the cost. In addition studies in this field can result in accessibility of telecommunication systems and internet in severe conditions. The potential social benefits that having high speed data transfer system available at various situation can provide, is also unlimited. The research in this field is directly derived by the demands of the society and economy. The deployment of researches in this field will bring vast economic benefits and can provide huge social benefits to the society. Our work in this field removes some of the obstacles in this path, answers a solution for some of the well-known problems, and in part opens new view point and research fields in the area of next-generation telecommunication systems.

For the hardware implementation of telecommunication system it is very important to

know what is the optimum performance of the system when precision is unlimited. We had to first investigate the performance of the system on such scenario and then find what is the minimum computation precision required for each element of the system such that the system performance is not reduced, and then report the cost of hardware implementation. We implemented the system simulator that models the performance of the system in AWGN channel using Simulink double precision floating point arithmetic (maximum performance) and then investigate to find the least complex hardware that requires minimum performance compromises while keeps the computation time within the system requirement. We performed the full procedure on the IEEE 802.16e [1] standard to ensure the simulation settings are reasonable.

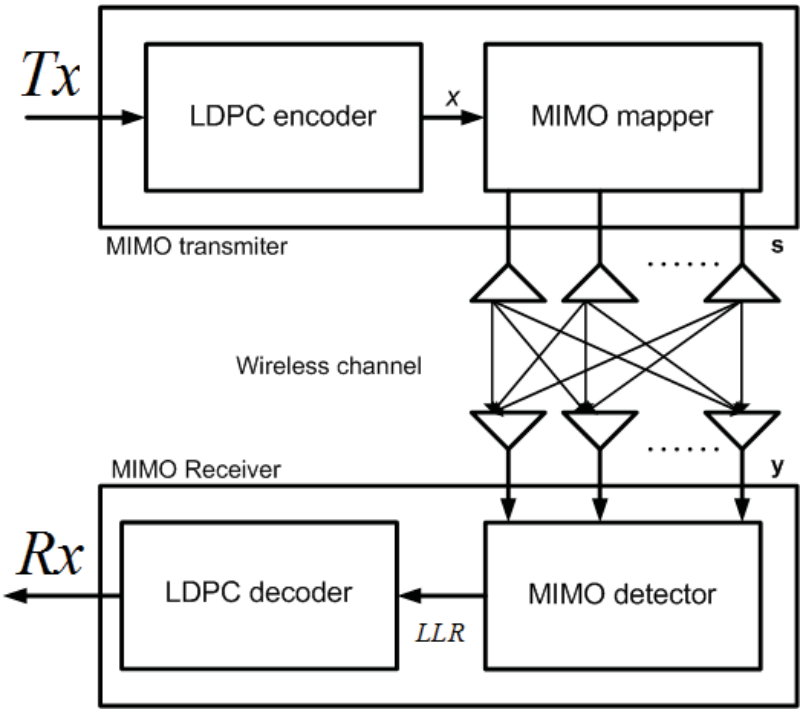


Figure 1.1: The block diagram view of MIMO transmitter and receiver.

1.2 Problem Overview

Figure 1.1 shows the block diagram of MIMO transmitter and receiver. This figure demonstrates the correlation between data transmitter on different antennas and signal received by the receiver antennas. One of the challenges in use of MIMO as a channel interface is to detect transmitted symbols. Assuming the channel matrix is **H** the received signal can be presented as:

$$\mathbf{y} = \mathbf{Hs} + \mathbf{n} \tag{1.1}$$

where **H** is an $M \times N$ channel matrix. The elements of **H** are constants during the transmission of a block of data (Block fading channel) and they are complex random numbers with average of zero and variance of one; **s** is the transmitted signal of $N \times 1$ dimension and **n** is a $M \times 1$ dimensional vector of additive complex symmetric Gaussian noise. The entries of **n** are chosen from a set of complex constellation of which the size is two to the power of transmit bits per symbol *(q)*. The total transmit bits per discrete transition is $M \times q$, where M is the number of transmit antenna. In this work we assumes 4×4 transmit and receive antenna arrangement unless otherwise mentioned. Also, we assume that the channel matrix and variance of noise are known to the receiver.

In this work we study the effective balance of the calculation complexity of the algorithm required for retrieval of the transmitted data and the error rate performance of recovered data at different channel conditions. This includes various trade-offs on the complexity of signal detection and decoding at the receiver.

1.2.1 Detectors

There are two type of detectors in general; the linear, and non-linear detectors. The non-linear detectors provide close to ML performance while in general their computational complexity is not only higher than that of linear detectors, but also it exponentially increases with the increase in constellation points or number of antennas. Linear detectors on the other hand, have compromised performance curve while their computational complexity is lower, and it increases linearly with increase in constellation points, and number of transmit or received antennas.

Non Linear Detectors

To solve Equation (1.1), one way is to exhaustively search for all possible constellations of transmitted symbols for the one that is closest to the received signal. This method can provide Maximum Likely-hood (ML) performance. However, this solution is very expensive in terms of hardware and energy cost. This exhaustive search then can be transformed to a tree search as follows:

$$Problem : \min\ d(\mathbf{s}) = \min \|\mathbf{y} - \mathbf{Hs}\|^2 \quad s.t. \quad \mathbf{s} \in \Omega \tag{1.2}$$

where, Ω is the symbol set of a modulation constellation. The equation shows a combination optimization problem to minimize the target equation by selecting the valid candidate from the modulation constellation.

The optimization problem is computationally exhaustive. We present QR decomposi-

tion here to convert this problem to a tree search problem.

$$\mathbf{H} = \mathbf{QR}, \quad where \;\; \mathbf{Q} \in \mathbb{C}^{M \times N}, \; \mathbf{R} \in \mathbb{C}^{N \times N} \tag{1.3}$$

In this equation $\mathbf{Q}$ is an orthogonal, and $\mathbf{R}$ an upper triangular matrix. Using the following steps the search algorithm will become a tree search problem:

$$\|\mathbf{y} - \mathbf{Hs}\|^2 = \|\mathbf{QQ}^T(\mathbf{y} - \mathbf{Hs})\|^2 + \|(\mathbf{I} - \mathbf{QQ}^T)(\mathbf{y} - \mathbf{Hs})\|^2 \tag{1.4}$$

$$= \|\mathbf{Q}^T\mathbf{y} - \mathbf{Rs}\|^2 + \|(\mathbf{I} - \mathbf{QQ}^T)\mathbf{y}\|^2$$

$$Problem : \min \; \tilde{d}(\mathbf{s}) = \min \|\tilde{\mathbf{y}} - \mathbf{Rs}\|^2 \quad s.t. \quad \mathbf{s} \in \Omega \tag{1.5}$$

$$where \quad \tilde{\mathbf{y}} \triangleq \mathbf{Q}^T\mathbf{y}$$

The problem can be presented as a tree search where each branch is one of possible transmitted symbols and the values on the nodes are the sum of all distances for the combination of transited symbol up to that level of tree. Figure 1.2 shows a tree search for BPSK and three transmit and receive antennas.

The goal of the tree search is to find a branch at the last layer of the tree that has the smallest $d(\mathbf{s})$ value on it. Although presenting the problem as a tree search can reduce the calculation complexity by reusing the values calculated in previous levels of the tree, it still requires exhaustive search. Another solution is to reduce the search area with one of the suboptimal search methods. Some detection algorithms based on the later method (such as sphere decoding) are able to provide close to ML performance with much smaller cost. Sphere decoding as shown in Equation (1.6) successively reduces the search space

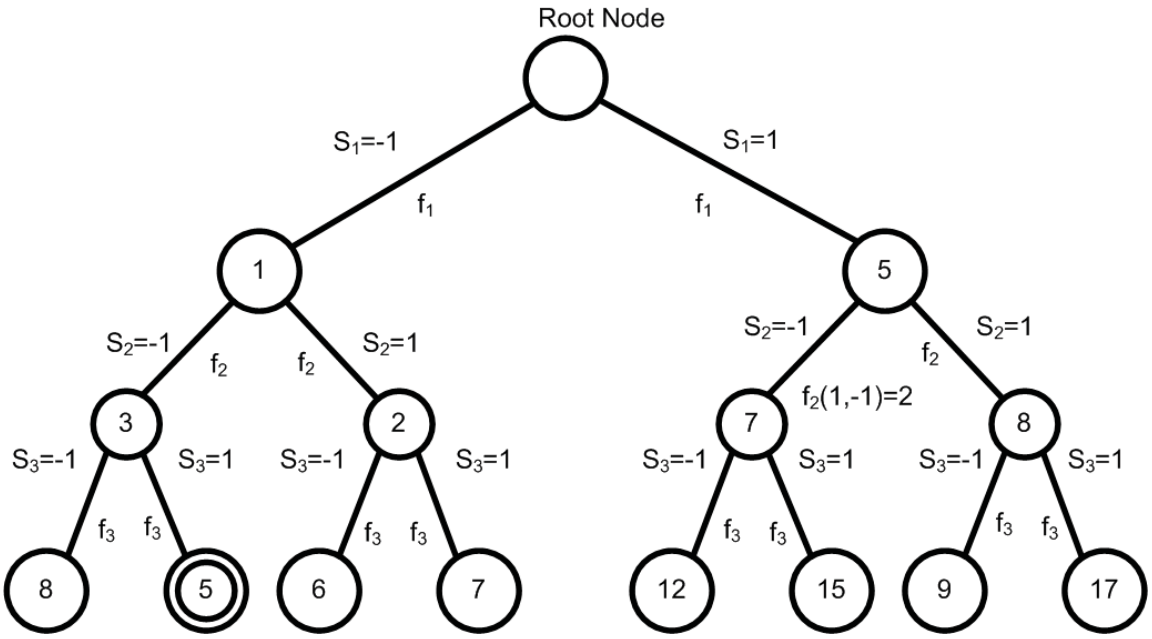


Figure 1.2: lattice search tree relevant to a 3x3 BPSK transiver.

by controlling the search radius (r).

$$Problem : \min\ \tilde{d}(\mathbf{s}) = \min \|\tilde{\mathbf{y}} - \mathbf{R}\mathbf{s}\|^2 : \tilde{d}(\mathbf{s}) < r^2 \tag{1.6}$$

Using sphere decoding, the detector needs only to evaluate those constellations that fit inside a sphere around the received signal. At any point if the cumulative sum of the values of the branches of the three becomes greater than the radius, that branch of the tree with all its lower level sub branches can be excluded from the search. Proper selection of the radius can affect the detection performance as well as the calculation complexity. In another word; while the Greedy algorithm includes all the possible candidate and the accuracy of its result is maximal, the demanding calculation complexity makes unviable option for power aware applications. On the other hand, the sphere detection algorithm demands less computational effort while if the radius is not set correctly could result in

performance loss (radius is set too small) or waste of calculation power (radius is set too large).

Linear Detectors

The detection procedure for linear receivers can be modeled as matrix **B** multiplied by the received signal, while each detector defines a unique **B** matrix. Then we can assume the detected signal as:

$$\mathbf{By} = \mathbf{BHx} + \mathbf{Bn} = \mathbf{Ax} + \mathbf{z} \tag{1.7}$$

where, $\mathbf{A} \triangleq \mathbf{BH}$ and $\mathbf{n} \triangleq \mathbf{Bz}$. In general, as the complexity of the algorithm to calculate the **B** increases the PER performance of that detection method improves.

1.2.2 Decoders

In this work we used low density parity check code (LDPC) as our block coding technique. This coding technique is being increasingly used due to their close to capacity performance (better than turbo codes for higher code rates), and low error floor. An LDPC code is define with matrix called H. Each row of the matrix is a parity check equation and columns are associated with received bits. Using Tanner graph the parity check equation is called check nodes and the coded bits can be present by variable nodes. A variable node is connected to a check node in case that the associated bit in H matrix is one. The process of decoding can be done with iteratively passing information trough the edges of the graph.

1.3 Main Contributions and Dissertation Organization

In section 2 an iterative MIMO detector hardware with close to optimum performance has been presented. Using iterative detection and decoding [2] in conjunction with list sphere decoding method in this design targets the use of MIMO in severe channel conditions [3]. This hardware has following abilities:

- Switching between different detectors is used in conjunction with iterative detection and decoding.
- The hardware and power costs and benefits of these methods are also provided and optimized to accommodate the low power requirement of the system.
- Use of Clock-gating and Power-gating to eliminate the power usage of the subsystems that are not active.
- Different number of outer iterations based on the channel SNR to gain acceptable error rate.

In section 3 the software implementation and complexity of Integer Forcing (IF) detector based on the slowest descent algorithm presented in [4] for IEEE 802.16e [1] standard is studied and the list of the contributions are as follows:

- Two low complexity soft demapping algorithms have been presented for IF detector.
- The testbed we used to evaluate the performance of the system is an existing standard (IEEE 802.16e [1]) without enforcing any changes to the current definition of the standard.

- Finding the minimum value for parameters of algorithm in [4] for current setup (without degradation in detector's performance).
- Slight changes in slowest descent algorithm presented in [4] to achieve lower computational complexity and higher detection speed.
- Complexity analysis and comparisons of IF detector with some classic detectors.
- calculating the precision required to perform each task in detector without degradation in detector's performance.

While researching on hardware implementation of IF detectors our research on singular value decomposer (SVD) [5] result in significant improvement on the performance of this unit and we filed a disclosure of an invention with the TAMU TEES department. To keep the integrity of the dissertation while concealing that section due to confidential information contained and also to emphasize the importance of the results achieved in this part of research we dedicated a separate section to SVD and its implementation (section 4). Section 4 summary of the contributions are:

- The hardware is implemented using fixed point arithmetic unlike other approximate rotation based algorithms [6], [7].
- The algorithm provided that is able to generate the singular value decomposition values for non-symmetric matrices.
- The proposed algorithm provides the "Normalized" values.

We extend the SVD hardware design presented in previous section to a general purpose processor that is able to perform The slowest decent algorithm. The summary of contribu-

tions are as followed:

- Special purpose processor has been designed.
- The processor is implemented using fixed point arithmetic.
- Hardware and power cost of this processor is reported.

In section 6 as the proof-of-concept [8] an asynchronous complementary logic design for MIMO receiver is studied and the results shows that considerable improvement can be achieved against charged particles effect (600 times the minimum charge required to logically flip a gate output).

2 LOW-POWER ON-THE-FLY RECONFIGURABLE ITERATIVE MIMO DETECTION AND LDPC DECODING DESIGN*

This section presents a low-power hardware design for MIMO detection that uses varying number of detection and decoding iterations and switchable detection methods to provide prespecified minimum error coverage at minimum power levels. The design is illustrated with the three different multiple-input and multiple-output (MIMO) detection algorithms; optimal detection subsystem is selected on-the-fly for a given Signal to Noise Ratio (SNR) level of a channel to maximize gain-over-power factor. Also, the receiver adaptively changes the number of detection/detector (outer) loop iteration to arrive at the best power-gain point. This approach yields a receiver design with a significantly reduced power at the cost of increased total silicon area. We illustrate the design with over 80% reduction in detection power for an application with a minimum quality of service (QoS) in SNRs as low as 8.7 dB (Eb/No) without compromising the throughput.

2.1 Introduction

The MIMO systems are well known for their ability to provide higher data rate in a given channel. These systems can increase the channel capacity or reliability with no cost of frequency spectrum. There are many new standards that use MIMO as their link between

*Reprinted with permission from "Low-power on-the-fly reconfigurable iterative MIMO detection and LDPC decoding design" by Ehsan Rohani, Jing Wei Xu, Gwan Choi, Mi Lu, 2014. Applied Mechanics and Materials, 496-500, 1825-1829, Copyright 2014 by Trans Tech Publications Ltd.

transmitter and receiver; 802.16e, HSDPA, 802.11n and 802.20. Most of these standards have a specified minimum error rate to guarantee QoS. For different standards based on the standard primary usage and intuitive specification the QoS is defined as an exact value in bit error rate (BER) or packet error rate (PER). DVB-T/H defines a coded BER of$2x10-4$ after the inner FEC decoder, this makes sure that the output bits after outer decoder are ``quasi-error free (QEF)". In IEEE 802.16e standard, $10^{-}6$ is specified as the minimum required BER. IEEE 802.11 a/g/n defines the PER of 0.1 as acceptable QoS [9]. In This content, reaching performances beyond the required minimum error rate seems not only unnecessary but also wasteful [10].

The main challenge in use of MIMO systems is the computation power and complexity necessary for detection of received signal in the receiver. Some feasible maximum-likelihood (ML) detector methods exists which use exhaustive search [11, 12] for QPSK modulation and 4x4 antenna arrangement. With increase in the number of transmit and received antenna and the number of bits in modulation, the complexity becomes overwhelming. The suboptimal detectors have been shown to have near-ML performance in these situations with acceptable complexity.

Suboptimal detectors have been extended further to provide soft-output detectors and soft input-soft output (SISO) detectors that are suitable for subsequent iterative decoding [13, 14]. Researchers further combined these SISO detectors with low density parity check (LDPC) decoder to feed the output result of LDPC decoder back to detector and update the soft values of the detector to get better performance; this is what we call it iterative

detection [2] and we call this the outer loop in contrast with LDPC iterative decoding which we call it inner loop. Basically the iterative decoding is a cycle between detector and decoder which uses the output of decoder to calculate the LLR of the same received symbol more accurately. The reason that motivates us to choose LDPC as the decoder in our iterative decoding is their near Shannon performance and its computational simplicity compare to other near Shannon decoders [15--17].

With the vast coverage range of new standards, different working environment of these communication systems and the mobility of the subscriber station (SS) the change in quality of the received signal is very high. It means the designed system may work in extremely varying conditions that small or large amount of effort is needed for detection of transmitted signals. Also, with everyday decreasing of the silicon cost it is reasonable to instantiate multiple detectors on a chip and select the proper one based on the system demands and channel conditions. Of course a system always can use the fastest algorithm with the best performance or it can use an algorithm with different levels of parallelization or pipelining to reach a certain throughput, but these solutions are not power aware and with limited amount of power supply like the case of any portable device a more cautious design is necessary.

In [10] and [18] the idea of having more than one detector to reduce the calculation complexity based on the channel SNR has been theoretically explored. In [18] the authors use zero forcing (ZF) and sphere search (SS) algorithm to deal with the changes in SNR. Although the detectors provide the soft output, but iterative detection has not been

considered in this work. For the measurement the calculation complexity and number of visited nodes has been considered. The system in [18] calculates the average SNR of each sub-channel and uses different detectors on that sub-channel based on the calculated SNR.

The novelty of this section is that switching between different detectors is used in conjunction with iterative detection and decoding. This means that one of the detectors in this system is SISO detectors which enable the system not only to choose between the detection methods but also to choose the number of outer loop to be applied on the received signal. The hardware and power costs and benefits of these methods are also provided and optimized to accommodate the low power requirement of the system. The architecture of the system uses Clock gating and power gating to eliminate the power usage of the subsystems that are not active. We used different number of outer iterations based on the channel SNR to gain acceptable error rate. Different performance for different algorithms has been observed in term of power consumption and hardware complexity in constant throughput. The system is able to measure the channel SNR and compare that with the threshold values calculated from simulations and base on that it will decide which detector to be used and what is the number of detection and decoding iterations.

Rest of this section organized as follow: First we will introduce the MIMO detection problem. Next we will discuss the basics of iterative detection. Later we will explain hardware of the detectors that we considered. We will discuss the LDPC decoder of our choice and the details of our case study. Then we will see the global architecture of the design and we will discuss the results. Finally we conclude the section.

2.2 MIMO Detection

The overall picture of a MIMO system is presented in Figure 2.1 In this section we introduce the problem of MIMO detection mathematically and we will show how this will lead to a tree search problem.

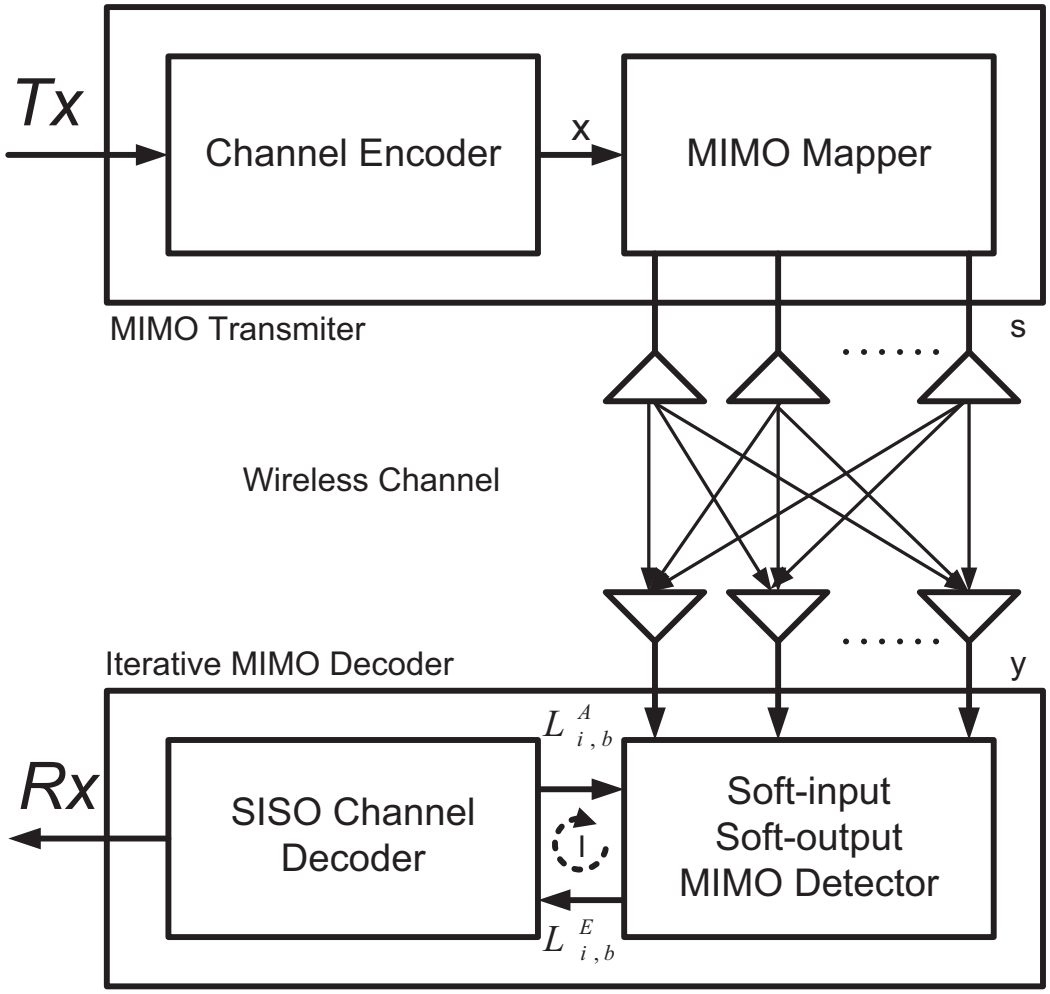


Figure 2.1: Block diagram of iterative decoding

Assume that the transmitted signal is s which is consisting of N symbols then the re-

ceived signal will be as follow:

$$y = Hs + e \tag{2.1}$$

H is a 2M by 2N matrix which M is the number of received and N is the number of transmit antennas. The elements of H are constants during the transmission of a block of data (Block fading channel) and they are complex random numbers with average of zero and variance of one and e is a Mx1 dimensional vector of additive complex symmetric Gaussian Noise. The entries of s are chosen from a set of complex constellation of Ω of which the size is two to the power of transmit bits per symbol (Q). The total transmit bits per discrete transition is T=NxQ.

For detection of s in maximum likely hood sense we can write the problem as:

$$Problem : Min\|y - Hs\|^2 \tag{2.2}$$

There are different solution approaches for Eq. 2 and one obvious solution is exhaustively searching all the possible transmitted streams combinations (all combinations of at all antennas Ω). However, sphere decoding as shown in Eq. 3 successively reduces the search space by evaluating only those constellations that fit inside a sphere around the received signal

$$Problem : Min\|y - Hs\|^2 : d(s) < r^2 \tag{2.3}$$

Using QL decomposition on Eq. 3 we will change the problem to a tree search problem. If R stands for Real numbers and r for radios of search sphere, then:

$$H = QL \tag{2.4}$$

$$Q \in R^{2M \times 2N}. \tag{2.5}$$

$$L \in R^{2N \times 2N}. \tag{2.6}$$

$$\|y - H_S\|^2 = \left\|QQ^T (y - Hs)\right\|^2 + \left\|\left(I - QQ^T\right)(y - Hs)\right\|^2. \tag{2.7}$$

$$\|y - H_S\|^2 = \left\|Q^T y - Ls\right\|^2 + \left\|(I - QQ^T)y\right\|^2. \tag{2.8}$$

Problem can be written as:

$$Problem: \ d(s) = Min \left|\left|Q^T y - Ls\right|\right|^2 : \ d(s) < r^2. \tag{2.9}$$

In other word:

$$\tilde{y} \equiv Q^T y \Rightarrow Problem: \ d(s) = Min \left|\left|\tilde{y} - Ls\right|\right|^2 : \ d(s) < r^2. \tag{2.10}$$

Knowing that L is a left sided matrix which its entire element right to the main diagonal is zero; it can be seen that the search for the result is basically a tree search problem. The tree has a root and the root has branches equal to *size* (Ω). Then each branch can be divided to new branches as we did for the root and this will continue for the number of transmit antenna times.

Figure 2.2 shows a tree that should be searched for BPSK and three transmit and receive antenna. The goal of the tree search is to find a branch at the last layer of the tree which has the smallest value on it. In continue we will discuss how soft values can be calculated and used.

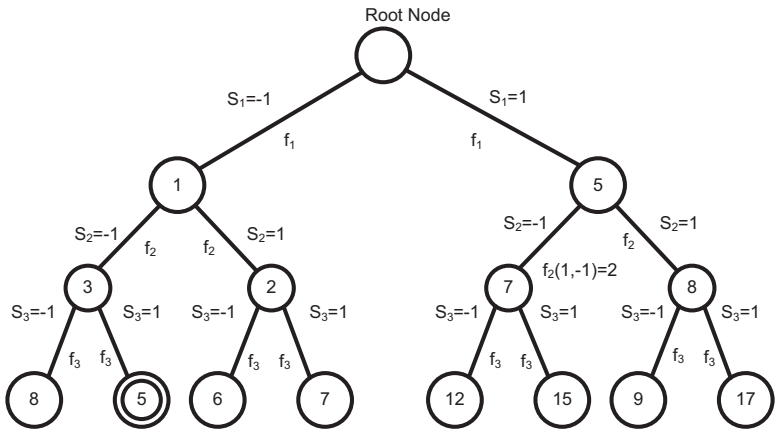


Figure 2.2: The lattice search tree of a 3x3 BPSK arrangement

2.3 SISO for Outer Loop

For iterative detection and decoding of the MIMO system, it is necessary to implement a soft input-soft output detector and decoder. LDPC soft decoder was implemented more than 12 years ago [19]. For detectors, Hochwald et al. in [15] described a method to efficiently calculate the approximate LLRs from a list of candidates. Using Eq. 12 makes it possible to implement a SISO detector. The task of generating and updating the LLRs will be done with a block called LLR update unit (LLR computation unit in case of MMSE-PIC).

$$L_E(x_k \mid y) = \left[\frac{1}{2} \max_{x \in X_{k,\ 1}} \left\{ \frac{1}{\sigma^2} \| y - H.s \|^2 + X_{[k]}^T . L_{A,[k]} \right\} \right.$$

$$\left. -\frac{1}{2} \max_{x \in X_{k,\ -1}} \left\{ \frac{1}{\sigma^2} \| y - H.s \|^2 + X_{[k]}^T . L_{A,[k]} \right\} \right] \tag{2.11}$$

Where $X_{[k]}^{T}$ and $L_{A,[k]}$ are the candidates values (-1 or 1) and LLR values but with the elimination of kth candidate. The LLR values at the output of LDPC decoder can be used as hard detection or can be feed back again to LLR update unit. Calculating Eq. 12 from a limited set of candidate instead of Ω creates the over confidence LLR problem which is the result of lack of hypothesis in calculation of Eq. 12 (The simulation results shows, using list size of 32 with limiting the LLR to ± 8 can mitigate this problem). The MMSE-PIC instead uses the statistic analysis to calculate Eq. 12.

The research in [13] shows that iterative decoding can increase the performance by several dB. While it is not the case here because the code length in [13] is almost ten times the code length for IEEE 802.11n; the MMSE and DFS-LSD detectors gain 2.2 dB and 1.1 dB compared to their first iteration, using only three detection and decoding iterations.

2.4 Detectors

In this section we consider three different detectors. These three detection methods which we have considered to employ with different channel SNRs are a). Depth first search list sphere decoding (DFS LSD) which generates a list of best candidates in first iteration. In second iteration (and iterations after that) it uses the candidate list and feedback values to calculate and update the LLR, b). Minimum mean-squared error based parallel interference cancellation algorithm (MMSE-PIC), and c). Layered orthogonal lattice detector (LORD); which have major differences in term of power usage, hardware cost and performance. This detector is less complicated in compare to two other detectors and it provides acceptable

performance. We used this detector because in compare with the other low complexity soft output detectors this is among the best in term of performance.

Not only have these three detectors had differences, in terms of implementation costs and power consumptions, they also vary considerably in term of gain when coupled with iterative decoding. In this section we describe the details of three different detectors we considered for our design and then we focus on the implementation issues and performance results of aforementioned detectors.

2.4.1 MMSE-PIC.

The first detector is MMSE-PIC. We used the architecture in [20]. This detector has less complexity in comparison to ML detectors and consumes less power. But this detector uses the maximum hardware among our detectors when it adopted to perform three outer iterations (3x259 KGE). Figure 2.3 shows the block diagram of the MMSE-PIC detector, you can see how the feedback LLRs go through the entire calculation blocks. We should mention that, using MMSE-PIC we obtain almost 1.5 dB gain in second iteration compare to the first iteration.

The power consumption of the detector is 25 mW for each outer iteration. As we will describe thoroughly later, although using MMSE-PIC and third outer iteration we were able to achieve 0.3 dB performance gain we decide that implementing an expensive detector (in terms of power and hardware costs) is not optimum in our case study. As this detector will not be present in our final system we avoid providing more details.

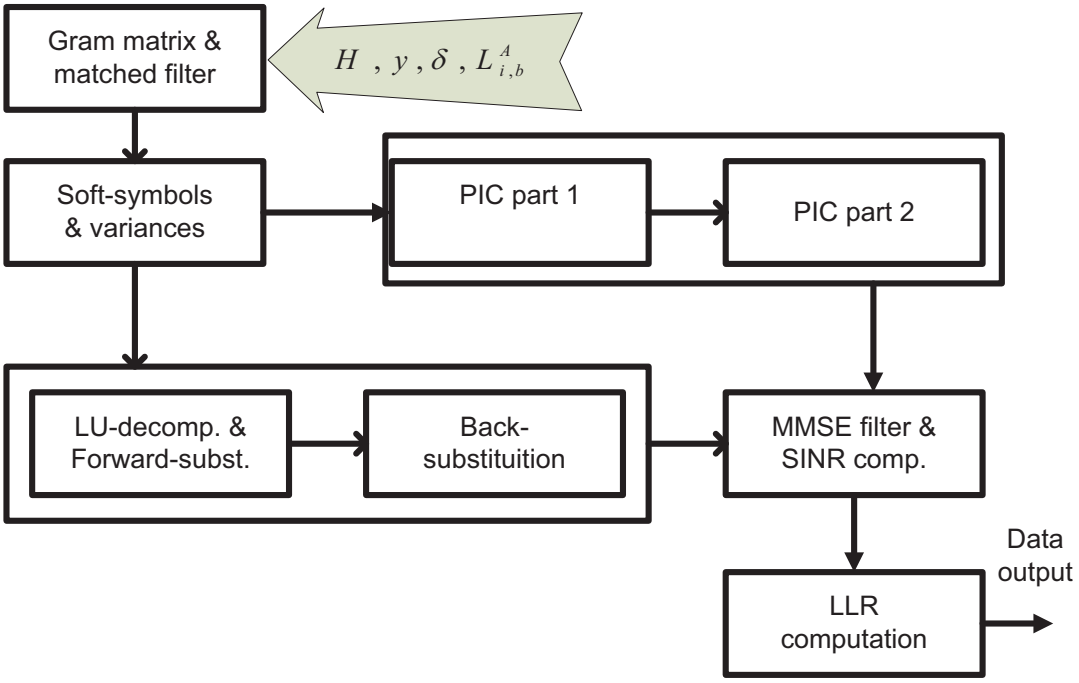


Figure 2.3: The block diagram of MMSE-PIC

2.4.2 List Sphere Detector

This category of the detectors search the lattice tree only once and they build a list of the strongest candidate for each received symbol then they will generate the LLRs and will update them using the candidate list and relayed calculated distances. In this work we used list sphere decoding with initial radius set to infinity in which the list size controls the effective radius directly. In this method a candidate that has cumulative value less than or the list radius is replaced with the candidate in the list with highest cumulative sum of values. The radius of the list is determined by the highest candidate cumulative sum of branch values. The list size directly affect the complexity of the tree search procedure and also, affects the complexity of LLR calculation and limiting the list size could cause

inaccurate LLR calculation. The task of generating and updating the LLRs will be done with a blocks called LLR calculation (update) unit. Most of the times these type of detectors even don't search the entire tree and they only search for their candidates within a certain distance of received signal (sphere). This is why they have been called sphere detectors.

Interesting point about this detector is that for iterations after first iteration the feedback LLRs will go to LLR update unit and using the pre calculated distances we are able to update the LLRs. This means that there is no need of other blocks to be presented and they can be turned off. As the result, although the LSD uses more power for first iteration in compare to MMSE-PIC but, in second iteration and iterations after that it will be more power efficient. It worth to mention that, using LSD we obtain almost 0.8 dB gain in second and 1.1 dB in third compare to the first iteration. The power consumption of the detector is 36.2 mW, 48.8 mW, and 61.1 mW for first, second, and third iterations. In following section we will describe the architecture of the two most important units of the LSD-DFS detector.

Tree Search Unit

We used the depth first search architecture presented in [21] as our List detector. In order for presented architecture in [21] to support list detector the tree search unit needs the following provisions.

The tree search keeps the K best distances at the last level

As a new node with less distance is reached, the search radius is not necessarily de-

creased unless all the candidates of the list have shorter distances. Thus we do not limit the radius at the beginning, and r in Eq. 11 initially will be infinity.

One of the important parameters of the LSD is list size. It has been observed that the performance of this kind of detector is highly related to the list size. If we decide the list size too large the power consumption will be too high and if we chose it to be small then the degradation in performance will be high. In another word, if we choose a list size which is twice the optimum size then the LLR update unit will almost use twice the optimum power to provide the same throughput. In [22] the LLR clipping has been used to reduce the list size effectively. The maximum list size which is beneficiary for this architecture is 32 which enable us to gain maximum possible performance out of iterative decoding.

The architecture of LSD is presented in Figure 2.4. The sphere ALU calculates all the PEDs in complex form. Discussion about using complex coefficients instead of real coefficient is presented in [23].

The basic difference of our presented architecture with the reference architecture is on last two blocks of MCU unit. In case a leaf has been reached, instead of keeping only the best result we keep a list of candidates. The MIN-search block decides the best list from the previous candidates and the new children. For list size of 32 and 16 QAM modulation, the search selects best 32 out of 48. Also when a leaf reached, we update the radius with the distance value of the weakest candidate among last 32 candidates in the list.

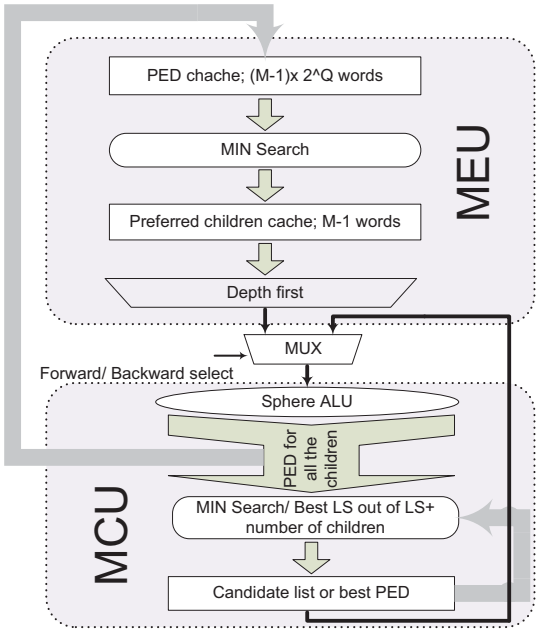


Figure 2.4: Architecture of tree search unit

LLR Update Unit

This unit consists of 16 adders and computes Eq. 12 based on the input distances and LLRs. The update unit can update the value of Eq.12 one time in each clock cycle. It takes one of the candidates at each clock interval and computes the LLR value, then the new LLR will be compared to the maximum of previous LLRs and the total maximum of entries will reach the last level of comparators. The update unit should keep track of two numbers for each LLR; one for those that the k^{th} bit of candidate list is one (Lambda-ML) and the other for zero (Lambda-ML-bar). After LLR update unit does the same process for all candidates of the list, the LLR values will be calculated as the subtraction of Lambda-ML and Lambda-ML-bar divided by two.

In Figure 2.5 the architecture to calculate Eq. 12 is presented. We tried to minimize the hardware cost of the architecture. The blocks at the first layer of the circuit is basically an XOR which calculates the one's compliment of input LLRs in case that the associated bit in selected candidate is 0 (-1). For the calculation of two's compliment the carry input of each adder is associated with one of the bits of candidates. Saturation blocks prevent from the overflow and the register are inserted to pipeline the architecture and increase the throughput. The last part of the circuit computes the max of Lambda-ML and Lambda-Ml-bar.

The delay of the architecture in Figure 2.5 is seven clock cycles. The rate of processing is one clock per candidate per LLR. Based on these facts, for a candidate list of 32 we need 32 blocks of this architecture working in parallel as the basic blocks of LLR calculation unit with 31extra comparators to calculate the max of Lambda-ML and 31 extra comparators to calculate the Lambda-ML-bar. Another way of doing this is to increase the frequency and using the architecture of Figure 2.5 as the LLR update unit. Targeting the throughput of 480 Mbps the clock should be 15.36 GHz. We used eight basic blocks of Figure 2.5 working in parallel in our system to create our LLR update unit.

As the input LLRs are zeros for first iteration, the necessary architecture of the update unit will be limited to the blocks presented after the last adder. The presented architecture takes seven clock cycles to calculate Eq. 12 for one candidate in the list and we have eight of this circuit running in parallel. This means that signals need to go through each of the circuits four times before being combined on all eight circuits. It takes one clock cycle to

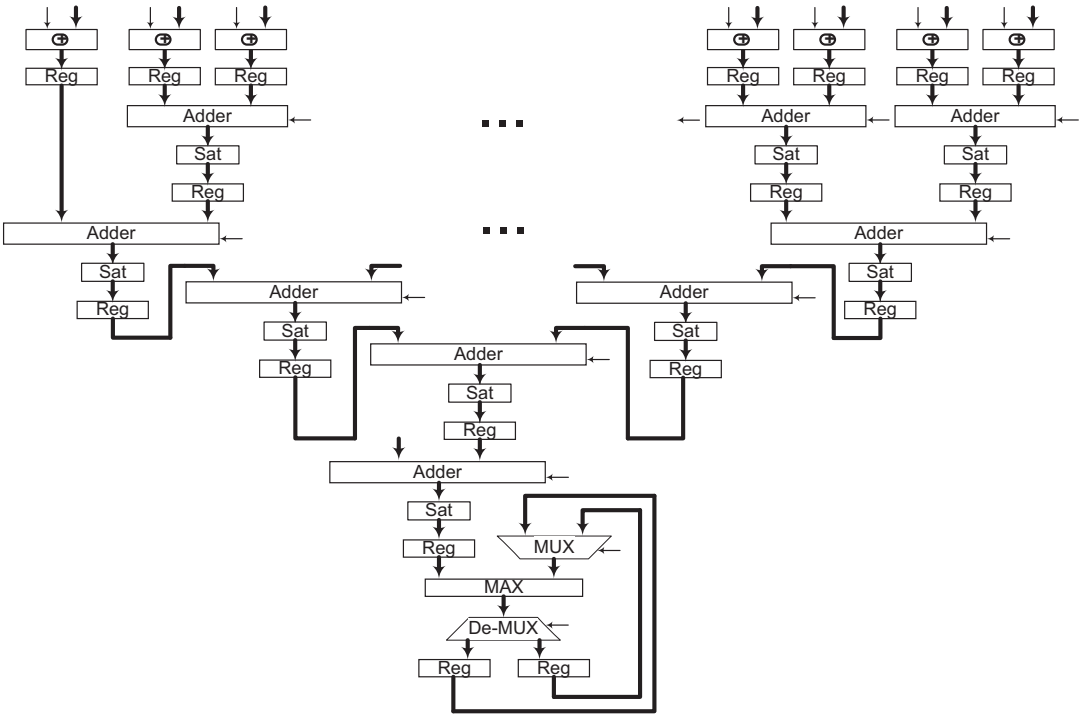


Figure 2.5: Basic block of LLR update unit

calculate the final value of LLR. This means the latency of the circuit is 28 clock cycles which is equal to 14 ns. The LLR update unit is completely parallelizable and we used this property in our benefit to provide a constant throughput architecture which works at 480 Mbps.

2.4.3 Layered Orthogonal Lattice Detector (LORD)

The third detector algorithm which we are using is LORD. The LORD algorithm [24] takes a suboptimal approach in searching for the two nodes that are required to compute the LLR value for a particular bit.

In [24], the channel matrix and the received symbol are preprocessed by real valued

decomposition (RVD) so that all the variables are converted into real numbers. Notice that after complex to real conversion, the dimensionality of all vectors and matrices in equation (2.1) will double. This implies that the tree search depth will also be double. An alternative way to do LORD is proposed in one of our previous works [25], which directly does the tree search without RVD. This is the architecture that we have used for the third detector.

As mentioned previously, MIMO detection is preceded by QRD. By permuting the columns of the H matrix, we can precisely control which QAM symbols will be processed first. LORD algorithm can compute the LLRs of the bits corresponding at the top level. To get LLRs in the below levels, it needs recomputed the QRD with columns-permuted H matrix. Apparently, to get all LLRs, QRD needs to be performed M times. A high level architecture of the LORD algorithm is shown in the Figure 2.6.

The presented Architecture is consists of an array of Metric Computer Units (MCUs) [25]. It can process four candidates in parallel. The Modulation Format (MF) information is fed into the control logic and MCUs to control the flow. Each MCU computes partial metrics according to Eq. 11. For a 4x4, 16QAM MIMO symbols, there are totally 4x16 candidate MIMO symbols to be evaluated. It takes 16 clock cycles to compute LLR values. The control unit is a simple finite state machine which takes in MF[1:0], which indicates the modulation format.

The matrices corresponding to symbol in each level will be stored in the LLR Computation Unit (LCU). The LLRs for all bits will be computed by LCU when the entire MIMO symbol is detected. The power usage of this architecture is 12.4 mW for throughput of 480

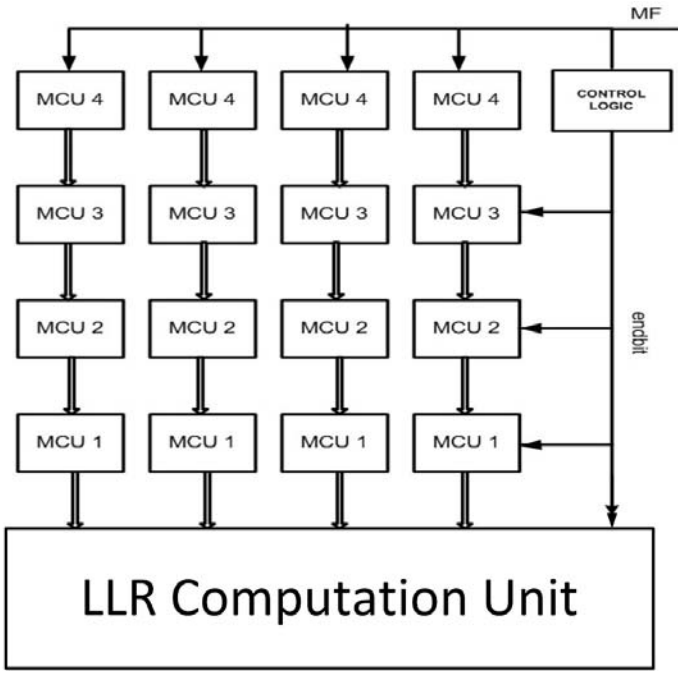


Figure 2.6: High level architecture of the LORD algorithm

Mbps.

2.5 LDPC Decoder

An LDPC code is define with matrix called H. Each row of the matrix is a parity check equation and columns are associated with received bits. Using Tanner graph the parity check equation is called check nodes and the coded bits can be present by variable nodes. A variable node is connected to a check node in case that the associated bit in H matrix is one. The process of decoding can be done with iteratively passing information trough the edges of the graph.

For the iterative decoder system presented in this section, we implemented the LDPC

decoder based on our previous work presented in [16]. The decoder accommodates irregular decoding with throughput of 480 Mbps and uses 19.7 mW, in 45 nm technology to perform maximum of 25 LDPC iterations.

LDPC decoder is allowed to iterate up to maximum of 25 cycles. For LDPC architecture this means 33 clock cycles for first iteration and 19 clock cycles for subsequent iterations. During three outer iterations, the LDPC decoder will take 1467 clock cycles (4.52 μs). Typically the number of LDPC iterations is highly sensitive to channel SNR. On the average, the number of LDPC iterations is 23.36 for first and second outer-iterations at SNR of 7.5 dB, and it can be as low as 3.57 iterations for third and fourth outer-iteration at SNR of 10 dB.

2.6 Case Study

The case study of this section is for IEEE 802.11n standard modulation and coding Scheme (MCS) index of 27. This MCS has 4x4 t antenna arrangement with 16 QAM modulation. The rate of LDPC coding rate is ½ and codeword length of 1944. We assume that the channel matrix and variance of noise are known to the receiver. The result of this case study is well extendable to other standards and MCSs but this case has been selected to be an indicator of a real standard arrangement with moderate complexity that provides high data rate (240 Mbps).

The reason that we used this numbers is that we wanted to make sure that we are aiming

at a reasonable complexity. In another word if we use lower modulation orders like QPSK the complexity of the detector will be feasible still with exhaustive search. Also the 4x4 transmit and receive antenna is the maximum limit for some standards that use MIMO. The packet error rate (PER) shall be less than 10%, for a PSDU length of 4096 octets.

In Figure 2.7 the simulation of all the possible regimes is demonstrated. The first observation is that applying more than three outer iterations is not cost-effective. This is due to the fact that both of the iterative detectors are providing around 0.1 dB gain from third to fourth iteration of their own. The second observation is that the second iterations of MMSE-PIC and LSD are providing almost the same threshold SNR for PER of 0.1 and LSD provides better performance for higher SNRs as the power consumption of both regimes is almost the same (LSD performs slightly better) we decide to remove the second iteration of MMSE-PIC from list of possible regimes. The final observation is that the MMSE-PIC first iteration is less efficiency compare to LORD (in term of hardware cost, power and, performance) so this makes first iteration of MMSE-PIC useless in this system.

In Figure 2.8 the performance of the final beneficiary regimes is presented. These are the regimes that using them is reasonable in term of consumed power and performance. Figure 2.8 shows that using three iterations of MMSE-PIC will provide us almost 0.3 dB benefit in performance. As using all other regimes that use MMSE-PIC as detector is not an optimum solution we decide not to implement this detector in our final hardware simply because we are assuming that the gain that it provides does not worth the hardware cost.

We should mention that, although the performance benefit of MMSE-PIC is not justi-

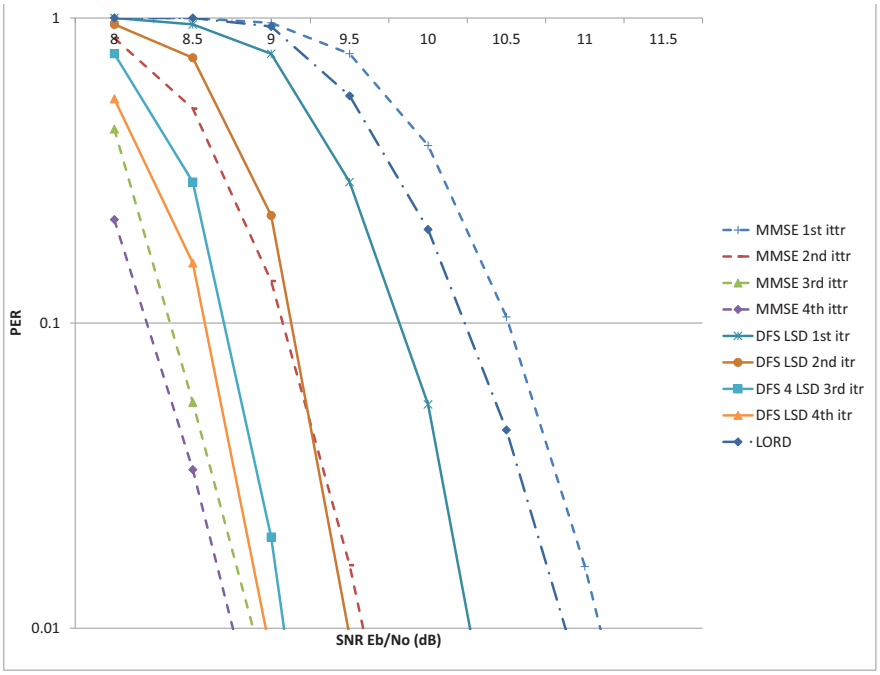


Figure 2.7: The simulation of the detectors

fying its hardware cost in authors' opinions in this case study, but using longer code length the performance benefit of third iteration will become grater and implementing this detector may become reasonable. Also; there may be cases that even this 0.3 dB performance gain justifies the hardware cost, if the silicon price is of less interest.

2.7 Architecture of the Design

The high level architecture of the design is presented in Figure 2.9 . We provide each of the detectors with a power supply that can be enabled or disabled. This can help the circuit to reduce the power consumption, considering that the buffers of the detector are also connected to the same power source as the detector. As the result of provided architecture

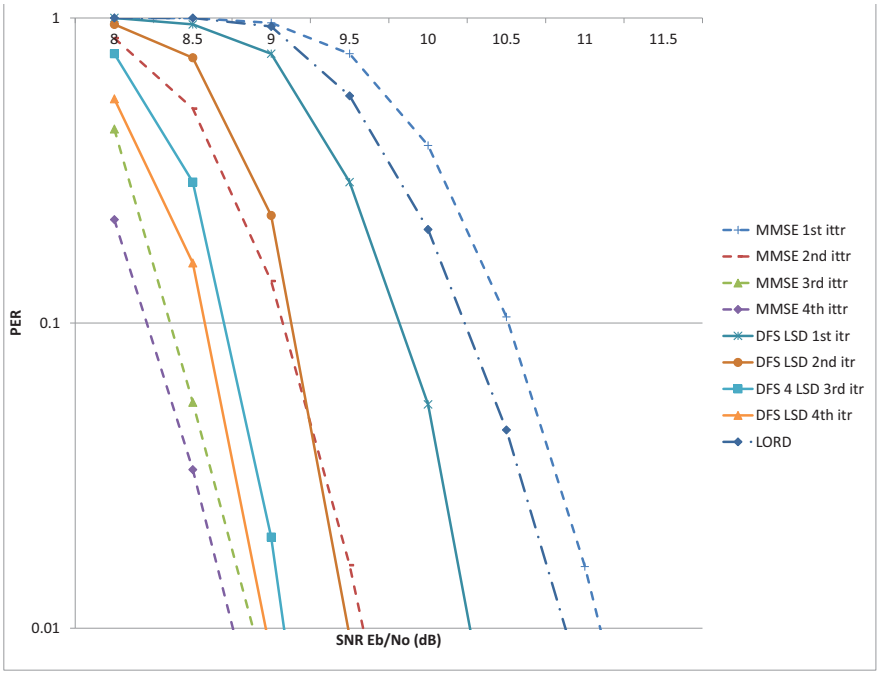


Figure 2.8: The beneficiary regimes.

the blocks which are not in use, will not be using power. We have connected the output of each detector to a multiplexer and we can select between the blocks output.

The architecture also uses clock gating which enables us to save power in boundary conditions. In case that the operating SNR of the system is in boundary range we power on the adjacent regime detector to pass initialization duration and we clock gate that detector. During a transition between one channel condition to another we simply turn on the power supply and after initialization time we will change the multiplexer to its new position at the beginning of the next received symbol. Then we will turn off the detector which is now offline. This technique makes sure that we do not lose any data during the transition period. This happens because at the margins both the detectors are able to provide an acceptable

detection of the received symbol.

As we discussed it previously we only implement two detectors in our final system which one of them (LORD) is not iterative. After decoding of data in the first iteration if the target regime has more than one iteration (which should be using DFS-LSD detector) the decoder output will feed to LLR update unit and using the list and distances calculated from first detection loop the LLRs will be updated. Each LLR update unit is provided with a separate power source and clock gating which enable the system to use them the same way as we discussed for detectors.

As displayed in Figure 2.9 we used three LDPC decoders in our design with throughput of 480 Mbps instead of using one decoder with throughput of 3x480 Mbps. The reason is that using this topology we are able to simply turn the unused decoders off and save the power without dealing with DVFS to reduce the power usage of decoder when its throughput is not completely used. Also each of the LDPC blocks (except the first one which is always on) are provided with clock gating ability and a separate power source which makes it possible to turn off the decoder that is not in use. This also gives us the ability to manage the initialization mode properly.

2.8 Results

We implemented our architecture using 45 nm standard cells technology. As described before, we always choose the less power consumer regime among the regimes presented in

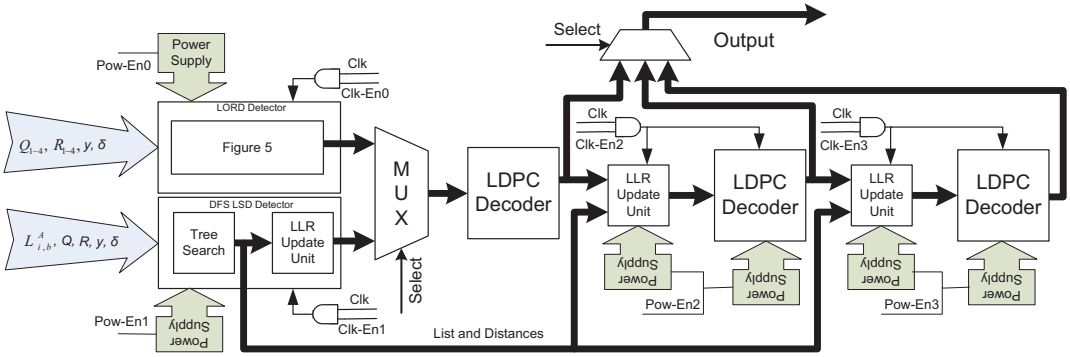


Figure 2.9: High level block diagram of the system

Figure 2.8. Table 1 shows the boundary values for regime changes. The marginal values are for AWGN channel and calculated based on the simulation of the system with the same parameters that we described earlier. These margins are the lowest SNR value for each of the beneficiary regimes that the QoS satisfy the requirement of standard.

Table 2 shows the hardware results and parameters of our detectors. The LSD detector uses 61.4 mW power for three detection iterations and 48.8 mW for two iterations. This means if the SNR of the channel is more than 9.1 dB we are able to save 20% of detection power and still be able to provide required QoS. This detector will use 36.2 mW for doing only one iteration of detection. As the result of switching ability of the system we will be able to turn the dedicated hardware for second iteration and third iterations off and save almost 26% in detection power compared to previous regime (two iterations of detection). Using this method we are able to save 41% detection power in total (compared with three iterations of detection).

In case that the SNR of channel is higher than 10.25 dB the system is able to switch to

Table 2.1: Margin SNRs of different regimes

SNR Range (dB)	8.7- 9.1	9.1- 9.8	9.8- 10.25	10.25- $+\infty$
Detector regime	DFS-LSD 3^{rd} iteration.	DFS-LSD 2^{nd} iteration.	DFS-LSD 1^{st} iteration.	LORD

LORD detector and at this point it will only use 12.4 mW. This will result in 66% power saving compared to previous regime. Using LORD in our system we reduced the detector power consumption by 80% and at any channel SNR greater than 8.7 dB we are satisfying the QoS condition for our IEEE 802.11n receiver.

The above reported power numbers assume only the detection power. Power reduction in both detection and decoding as the result of proposed switching technique are, 27%, 53%, and 73% (comparing power usage of each regime with using three iteration of LSD). If we compare power usage of each regime with power usage of its safer regime the power numbers are, 27%, 36%, and 42% (power that is saved).

2.9 Conclusion

In this section we considered three different types of MIMO detectors on a special case of IEEE 802.11n standard and we used two of the studied detectors in our subsequent iterative detector. Base on the properties of these three detectors we achieve different power, performance and complexity characteristics. These different characteristics gave us variety of choices which later were helpful to choose the optimum (minimum power usage while satisfying QoS) regime from the system available regimes.

Table 2.2: Implementation results of different Detectors

Detector	LORD [25]	DFS-LSD		
		Tree search	LLR calc. (1^{st} itr.)	LLR update (after 1^{st} itr.)
Antenna Conf.	4x4			
Modulation	16 QAM			
Performance	Near-ML	SISO iterative (1.6 dB gain to LORD)		
Technology (μm)	0.045	0.045		
Area (GE)	69K	209K	1.6K	30K
Throughput (Mbps)	480	480@ SNR 8.5 dB 580@ SNR 20 dB	480	480
Power (mW)	12.4	35.3	0.9	12.6
Latency (ns)	38	264	14	4

The global architecture of the system enables us to use the benefit of two implemented detectors with almost no redundant power dissipation. The system gives us ability to save up to 78% in power consumption of detection subsystem and 73% of power usage of detection and decoding without compromising the required performance for the system. This amount of saved power is really helpful especially when the power source is limited like cell phones or portable devices.

3 TWO STAGE SOFT-DETECTOR INTEGER FORCING SOFTWARE DEFINED RADIO FOR WIMAX USING SLOWEST-DESCENT METHOD

We present a software defined radio (SDR) solution for multiple-input multiple-output (MIMO) system using integer forcing (IF) receiver on a general propose Intel i7 processor. We introduce two low complexity soft demapping methods that are able to provide the approximate log likelihood ratios (LLRs) for QAM mapping in IF detectors. We also proposed changes to the algorithm that improves the system performance and reduces the required time complexity. In addition we compare the time complexity and packet error rate performance versus channel noise of the IF detector with zero forcing (ZF) and minimum min square of errors (MMSE) detectors[1] as well as the list sphere decoding depth first search (LSD-DFS) detector[2] in a 4×4 IID (independent identically distributed) MIMO channel and 16 QAM modulation testbed. In addition compare the changes in complexity with increase in number of antennas and higher order modulation. The required precision of IF detector is presented at the end.

[1] Linear detectors.

[2] Search based ML (maximum likelihood) detector.

3.1 Introduction

The advancements in the technology and increasing demand for higher speed in every subsequent generation of telecommunication system calls for massive improvements in throughput [26]. The requirements of a new generation is often met utilizing a new technique or using an older technique more efficiently and aggressively. This requires changes to standard specifications. Necessary changes are presented either as a revision (amendments) to an older standard [27] or they are molded into a new standard [28].

Each communication standard has set physical layer specifications (coverage range, data transfer throughput, physical environment[3], communication range[4], mobility [5],and frequencies) of its own but it also has overlaps with other standards and they should be able to coexist and cooperate. The coexistence and cooperative functionality of two different systems that are using same or different standards has been the focus of researchers. Cognitive radio is one of the solutions [29]. The ability of cognitive radio to reconfigure is mostly supported by their SDR that supports the system [30], [31]. Generally the reconfigurability of a cognitive radio is directly related to how sophisticated its SDR part is. Although SDR provides the reconfigure ability and flexibility it is generally slower (depends on the algorithm and the hardware being utilized) than the application specific integrated circuit (ASIC) solutions.

The advancements in the telecommunication technologies often translate to more com-

[3]Ex. line of sight or non line of sight.
[4]Physical distance between transmitter and receiver
[5]Movements of transmitter and/or receiver, specially the Doppler effect

putation time and power consumption in a specific time interval. An example of such advancement is use of the multi-input multi-output channels (MIMO) in recent years and the massive MIMO for 5G. MIMO uses multiple antennas at the transmitter and multiple antennas at the receiver in order to improve channel condition or increase the data transfer throughput. For detection of a transmitted signal in MIMO system some methods try to separate each stream of transmitted data and deal with it accordingly (linear receiver), while the optimum solution is to assume there is a correlation between streams of data and find the most probable sets of symbols (search based detectors). The complexity of a linear receiver, increases linearly with increase in the number of data streams and constellation points in a symbol while in ML detectors is exponential. Considering the fact that massive MIMO for 5G suggest the usage of upto eight data streams and 256 constellation in a symbol the required calculation power would be much higher and SDRs are sensitive to such changes since speed is not their forte.

The IF receivers are set of receivers that, can potentially provide higher performance than other linear receivers. As such, an IF detector is a proper candidate for a SDR based MIMO detector. IF detector tries to find an invertible (nonsingular) integer combination of transmitted symbols, ideally without increasing the noise. After detecting the signal an IF detector uses the invertibility and the fact that an integer combination of codewords from a linear code is also a codeword within the set to find the transmitted signal.

IF linear receivers can also be used for relay channels with a method called Compute-and-Forward [32]. In this method the nested lattices codes been used whose their algebraic

structure ensures that integer combinations of codewords can be decoded reliably. Encoders map messages from a finite field to a lattice and decoders recover equations of lattice points which are then mapped back to equations over the finite field.

IF detector however promising, has some drawbacks that we address in this work. Finding the likelihood of the received data bits is not as straightforward as other linear detectors with QAM modulations [6]. Using hard decoded bits is also not advisable specially since the gain achieved by soft decoding considered essential for the systems that require high quality in term of low error probability and automatic repetition of erroneous blocks can not be applied. "Therefore, very strong FEC[7] coding is a must" [33].[8] The two demapping technique that we proposed are soft decoding techniques that use approximate log likelihood ratio. Also; the mapping that is normally defined in current standards is not a nested lattice code. This potentially can prevent the use of IF method on any of the existing standards. We use the specifications of IEEE 802.16e standard for FEC and modulation mapping. In contrast with search based detectors, IF detector is eliminating the need for a search (within the poll of all possible combination of transmission symbols) per received symbol; however, in contrast with linear detectors, finding an invertible integer combination of transmitted symbols requires a search per transmitted packet. To retain the candor nature of IF linear detector we use the search method introduced in [4] as the solution.

[6] In such case the logarithm of the received signal distance from the nearest constellation in the modulation map (approximate log likelihood ratio) or from the center of the constellation map (log likelihood ratio) divide by the power of the noise directly translates to likelihood of the received symbol being detected correctly

[7] Forward error correction

[8] Ex. *deep space communication*

We provide the computation time and packet error rate (PER) performance of an IF linear receiver over a selected down link (DL) MIMO operation mode in an existing communication standard and compare them with those of other classic detectors as the reference for SDR solution. The IF results are compared with two other linear detectors, MMSE, ZF detectors. We compare the IF detector and ML detector performance in a similar situation, as well. The classic linear and search based detectors have been selected for comparison proposes due to their extensive presence in the literature and the fact that they provide a broad baseline for comparison. This work also provides two soft methods to avoid turning the demapping procedure to a search problem caused by incompatibility of the standard defined constellation mapping;[9] in addition it suggests the optimum parameter for the Slowest-Descent method [4] in the case study setup.

Rest of this section is organized as follow: We first present the MIMO and different detectors that we used in mathematical form and our assumptions in section 3.2. Next we discus the Slowest-Descent method in detail in section 3.3. We compare and evaluate our results in terms of performance and time complexity, in section 3.4. Finally we conclude this work in section5.6.

3.2 MIMO and Detection Methods

MIMO is a method of increasing the data transfer throughput using multiple antennas in transmitter and receiver. MIMO transmission over a channel can be presented as

[9] IF detector requires the constellation maps to be of a field with prime P which is not the case for WiMAX standard.

$\mathbf{y} = \mathbf{Hx} + \mathbf{z}$, where $\mathbf{x}$ is the transmitted signal matrix, $\mathbf{H}$ is the channel matrix, $\mathbf{y}$ is the received signal, and $\mathbf{z}$ is the noise matrix. In general, elements of $\mathbf{y}$, $\mathbf{H}$, $\mathbf{X}$, $\mathbf{N}$ are in $\mathbb{C}$; but, since using (3.1) any complex system matrix can be converted to equivalent real system matrix, we will use real presentation for convenience [34]. $\mathbf{H}$ is a $L \times N$ matrix where L is twice the number of received and N is twice the number of transmit antennas. Consequently $\mathbf{x}$, $\mathbf{z}$, $\mathbf{y}$ are $N \times 1$ matrices. We assume the receiver knows $\mathbf{H}$, $\mathbf{y}$, and the constellation mapping of $\mathbf{x}$. Also, in this work we assume $L = N = 8$ (equivalent of a 4×4 antenna arrangement),the constellation is 16QAM (quadratic amplitude modulation) with the peak power of P, and noise and channel matrix elements are independent and identically distributed (i.i.d.) random variables with average of zero and variance of $\sqrt{2}/2$ (this makes the variance of complex distribution is equal to one).

$$\begin{bmatrix} Re(\mathbf{y}) \\ Im(\mathbf{y}) \end{bmatrix} = \begin{bmatrix} Re(\mathbf{H}) & -Im(\mathbf{H}) \\ Im(\mathbf{H}) & Re(\mathbf{H}) \end{bmatrix} \begin{bmatrix} Re(\mathbf{x}) \\ Im(\mathbf{x}) \end{bmatrix} + \begin{bmatrix} Re(\mathbf{z}) \\ Im(\mathbf{z}) \end{bmatrix} \tag{3.1}$$

The detection procedure for linear receivers can be modeled as matrix $\mathbf{B}$ multiplied by the received signal, while each detector defines a unique $\mathbf{B}$ matrix. Then we can assume the detected signal is,

$$\mathbf{d} = \mathbf{By} = \mathbf{BHx} + \mathbf{Bz} = \mathbf{Ax} + \mathbf{n} \tag{3.2}$$

while, $\mathbf{A} \equiv \mathbf{BH}$ and $\mathbf{n} \equiv \mathbf{Bz}$. In general, as the complexity of the algorithm to calculate the $\mathbf{B}$ increases the PER performance of that detection method improves. While the ZF approach nullifies the interference between channels so that $\mathbf{A}_{ZF} = \mathbf{I}_L$ with the cost of increasing the noise power, the MMSE detector aims to maximize the post-detection signal-

to-interference plus noise ratio (SINR). The ZF and MMSE detectors calculate the **B** matrix as follows:

$$\mathbf{B}_{ZF} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T, \tag{3.3}$$

$$\mathbf{B}_{MMSE} = (\mathbf{H}^T\mathbf{H} + \frac{1}{P}\mathbf{I}_L)^{-1}\mathbf{H}^T \tag{3.4}$$

where $(.)^T$ denotes the matrix transpose operation and I_L is the identity matrix of size $L \times L$.

Search based detectors are more complex category of MIMO detectors. This category of detectors search a subset of possible space to find the transmission with highest possibility. The search has to be performed for each transmission. The ML detector in this category is a detector that tries to find the minimizing argument to minimize,

$$\underset{\widehat{\mathbf{x}}}{\operatorname{argmin}} |\mathbf{y} - \mathbf{H}\widehat{\mathbf{x}}|^2, \tag{3.5}$$

where, $|.|$ denotes the absolute function, and $\widehat{\mathbf{x}}$ is every possible combination of constellations.

While the search within all possible combination of transmit constellations make the ML detector expensive, the IF detector uses the basic property of lattice codes to find the linear receiver, ideally without increasing the noise. The IF detector recognizes the algebraic structure of lattice codes and its property that an integer combination of a lattice codeword is still a lattice codeword, and tries to find a full rank matrix **A** such that, all the elements of **A** are integer values, and also the achievable rate is maximized. In fact the IF detector allows a wider range for **A** matrix instead of restricting it to identity matrix (in case of ZF detector). The IF detector finds the **A** matrix using a search method but this

search has to be done only once for each transmitted packet since we assume the channel matrix is constant during a packet transmission (slow fading channel). Also, using a proper method [4] the search complexity of IF detector can be reduced effectively. After finding the $\mathbf{A}_{IF}$, (5.5) can be used to obtain the $\mathbf{B}_{IF}$. In next section we are going to present, evaluate, and improve the method of finding $\mathbf{A}_{IF}$ presented in [4].

$$\mathbf{B}_{IF} = \mathbf{A}_{IF}\mathbf{H}^T(\mathbf{H}\mathbf{H}^T + \frac{1}{P}\mathbf{I}_L)^{-1} \tag{3.6}$$

3.3 Algorithm Evaluations and Improvements

The reference [4] presents two algorithm that generate $\mathbf{A}_{IF}$ together. The *Algorithm 7* takes $\mathbf{Q}$ [10], J[11], and M[12] and generates a set of candidates Ω[13] using Slowest-Descent method. Using sorted ω and $\mathbf{Q}$, *Algorithm 8* constructs a full rank matrix $\mathbf{A}_{IF}$ that the total achievable rate is maximized. The second algorithm is slightly modified from [4]. The original algorithm does not mention in case the number of candidates are less than L and the A_{IF} is not achievable what should be substituted. This will result in loss of a complete packet which in fact will increase the BER (bit error rate) and PER (packet error rate); also, using the original algorithm even in very high Eb/No the PER and BER will have a error floor due to the lost packets (an unsuccessful construction of $\mathbf{B_{IF}}$ is additional error to error caused by channel noise and it should be considerably less than 1% or 0.1% if those PERs

[10] $Q \equiv A_{IF}H^T(HH^T + \frac{1}{P}I_L)^{-1}$
[11] Number of slowest descent lines.
[12] Bound for each coordinate of searching vector.
[13] Candidate set.

are target.[14])

The detection process has to be completed with demodulation after multiplication of B_{IF} with received signal and before decoding. The hard detection process is completed with finding the closest possible constellation to the received signal and then demapping the constellation to data. If the constellation is of lattice code then any integer combination of lattice codes will be a lattice code. While the standard constellations are not lattice code and this turns the detection to a search problem within the unique set of possibilities for each B_{IF} (upto CN^L, and CN is the number of constellations in the QAM[15] map). We present two sub-optimum soft detection methods that does not require search.

Algorithm 1 Candidate Set Search Algorithm with Slowest-Descent Method

Input: $\mathbf{Q}$, M, $(1 \leqslant J \leqslant L-1)$.
Output: Search vector candidate set Ω.
Step 1: Calculate the eigenvectors $\mathbf{g_1}, \mathbf{g_2}, ..., \mathbf{g_L}$ of Matrix $\mathbf{Q}$ with corresponding eigenvalues $\lambda_1 \leqslant \lambda_2 \leqslant ... \leqslant \lambda_L$.
Step 2: Compute the set of Λ:

$$\Lambda = \{m_j, \quad j = 1,2,...,2M+2\}$$
$$= \{-M - \frac{3}{2} + j, \quad j = 1,2,...,2M+2\}$$

Step 3: For $i = 2,...,J+1$,
(i) Obtain the jump points where:

$$\rho\prime_{(j-1)\times L+k} = \frac{m_j - g_{1k}}{g_{ik}}$$

with $k = 1,...,L, \quad j = 1,2,...,2M+2$.
(ii) Sort $\rho\prime_1, ..., \rho\prime_{(2M+2)\times L}$ to $\rho_1 \leqslant ... \leqslant \rho_T$
(iii) Find the closest point $\mathbf{a}_{(\rho_\mathbf{j}, \rho_{\mathbf{J+1}}]}, i$ as:

$$\mathbf{a}_{(\rho_\mathbf{j}, \rho_{\mathbf{J+1}}]}, i = round\left(\mathbf{g}_1 + \frac{1}{2}(\rho_j + \rho_{j+1})\mathbf{g_i}\right)$$

(iv) Clear the points that are not in range $[-M, M]$ in any coordinate, exclude the zero vectors, and the remaining vectors in Ω_{i-1}.
Step 3: Make $\Omega = \Omega_1 \cup ... \cup \Omega_J$.

[14]Unsuccessful construction of $\mathbf{B_{IF}}$ has severe effect on BER performance since it cause a burst of errors. in that case the probability of an unsuccessfully $\mathbf{B_{IF}}$ contraction should be zero which highlights the importance of change in algorithm 8 we made (Step 3).
[15]Quadrature amplitude modulation

Algorithm 2 Greedy Search Algorithm to Form $\mathbf{A_{IF}}$.

Input: $\mathbf{Q}$, Ω.
Output: Full rank matrix $\mathbf{A_{IF}}$.
Step 1: Let $f(\mathbf{t}) = \mathbf{t^T Q t}$ and sort all vectors in the searching candidate set Ω such that:

$$\Omega = \{\mathbf{t}_1, ..., \mathbf{t}_{|\Omega|} : f(\mathbf{t})_1 \leqslant ... \leqslant f(\mathbf{t}_{|\Omega|})\}$$

and set the $\mathbf{a}_1 = \mathbf{t}_1$, initialize $i = 1$, and $j = 1$.
Step 2: While $i < |\Omega|$ and $j < L$, do
(i) Set $i = i + 1$.
(ii) Check whether $\mathbf{t}_i, \mathbf{a}_1, ..., \mathbf{a}_j$ are linearly independent according to the determinant of the Gramian matrix. Let $\mathbf{G} = [\mathbf{t}_i, \mathbf{a}_1, ..., \mathbf{a}_j]$. If det($\mathbf{G^T G}$)$\neq 0$, then $j = j + 1$ and $\mathbf{a}_j = \mathbf{t}_i$, else goto **Step2**.
Step 3: If $j = L$, then $\mathbf{A_{IF}} = \mathbf{a}_1, ..., \mathbf{a}_L$, else $\mathbf{A_{IF}} = \mathbf{A_{MMSE}}$.

3.3.1 Detection Method One (DM1)

This method uses the properties of constellation and $\mathbf{A_{IF}}$ matrix and argues that if both of them are integer numbers then $\mathbf{A_{IF}x}$ also should be an integer. After rounding to the closest integer the result is multiplied by $\mathbf{A_{IF}^{-1}}$ and apply the soft demodulation (for decision type proximate likelihood ratio is been used). We note that if the rounding to nearest integer doe not take place the result would be same as a MMSE detector.

3.3.2 Detection Method Two (DM2)

The second method uses the same properties as DM1 except that it assumes each constellation is mapped to consequential integer numbers on real or imaginary axis (e.g. [0,3] for 16QAM). This will reduce the chance of rounding the received signal to an integer number that is not in the possible space. Using this method requires some extra mathematical

calculations. In order to use this detector the $\bar{Y}$ should be calculated using:

$$\bar{\mathbf{y}} = \frac{1}{2}(\mathbf{y} + \mathbf{H}C\mathbf{Ones_L}) \tag{3.7}$$

$$\bar{\mathbf{d}} = \mathbf{B_{IF}}\bar{\mathbf{y}} \tag{3.8}$$

$$\mathbf{d}' = 2 \times Round(\bar{\mathbf{d}}) \tag{3.9}$$

$$\hat{\mathbf{d}} = \mathbf{A_{IF}^{-1}}\mathbf{d}' - C\mathbf{Ones_L} \tag{3.10}$$

where, $\mathbf{Ones_L}$ is a $L \times 1$ matrix, and $C = 2 \times log_2^{(CN/2)} + 1$. The $\hat{\mathbf{d}}$ can be fed to a soft demodulator for final detection.

In the next section we will present the performance and computation time of different detectors using the same hardware and software environment. Although the previous versions of standard [35] used BER as their measure for minimum quality of service (QoS); the later standards use [27] PER as their target measure for ARQ/HARQ (automatic repeat request/hybrid ARQ). We presented the performance on the next section based on PER.

3.3.3 Comparison of DM1 and DM2

The comparison of DM1 and DM2 for both uncoded and soft-coded method is demonstrated in Fig. 3.1. The difference of soft detection is 5.5 dB at PER of 10^{-2} and 7.5 dB at PER of 10^{-3}. The Difference between two uncoded detection method is 9.5 dB and 11.5 dB respectively. Our complexity analysis shows less than 1% (0.357 ms for DM1 and 0.36 ms for DM2) difference between time complexity of two detection methods (The test conditions as described in section 3.4). Due to the performance difference of two method and proximity of their time complexity we will use DM2 for our analysis here after.

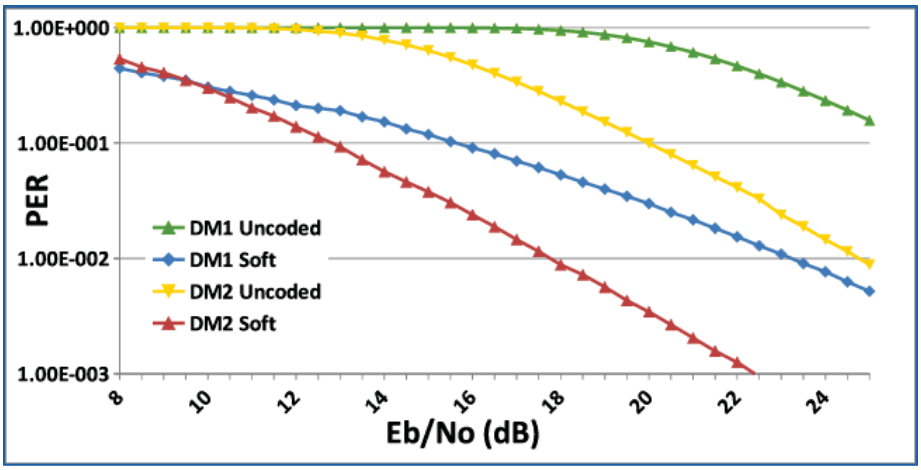


Figure 3.1: Computation of DM1 and DM2 Vs. Eb/No.

3.3.4 Reducing the Algorithm Complexity

The **Step 2** of the **Algorithm 8** is repetitive and calculation heavy. At each iteration the matrix needs to be multiplied by its transpose and then the determinant of the matrix needs to be calculated. If there is a dependency between the recent vector added to matrix and previous vectors, the most recent vector needs to be replaced. The chance of dependency between the vectors would increase with the increase in the number of vectors in the matrix. The complexity of calculating the matrix multiply and determinant also increases with the increase in number of vectors in matrix. If the size of the matrix increases due to increase in the number of transmit and receive antennas this would increase the complexity even more. The third column in Table 3.1 in shows the number of FLOPs required for calculation of matrix multiplication ($\mathbf{G}^{\mathbf{T}}\mathbf{G}$) with different number of vectors in the matrix. It also shows the average number of FLOPs required for calculation of determinant of different size matrices in the second column. The forth column shows the average (over 10^5 packets)

Table 3.1: Complexity (FLOPs) of Finding Independent Vectors Using Determinant.

# of Vect.	Avg. Det.	Avg. Mult.	Avg. Tries	Total
2	34.67	12	1.001	46.73
3	75	45	1.089	130.72
4	133.3	112	1.13	277.3
5	211.67	225	1.304	569.44
6	312	396	1.429	1011.6
7	436.33	637	2.196	2356.83
8	586.67	960	2.745	4245.5

number of tries required to find the independent vectors. The total number of FLOPs for finding an independent vector each time is calculated at last and it adds up to 8638.1 FLOPs.

Considering the fact that the search algorithm is already suboptimal we recommend another suboptimal search method with less complexity and same performance result. We form an 8×8 matrix using L first vectors of the Ω. After QR factorization; the matrix rank is calculated (number of the non zero diagonal elements of R); if the rank is less than L the vectors that generated zero diagonal element should be removed and instead next vectors of list (Ω) should be appended to the matrix. The new search algorithm is presented in **Algorithm 3**. The cost of calculating the QR factorization of a randomly generated dense matrix of size eight is equal to 1024 FLOPs. The average number of QR decomposition required (calculate over 10^5 packets) is equal to 3.9674. The average number of FLOPs required to form the A_{IF} matrix is 4062.62. The calculation complexity of This Step in new algorithm is less than half the complexity of original algorithm. AS we will demonstrate in next section this technique will reduce the complexity of **Step 2** to less than $\frac{1}{3}$ of the original method. This method however will not change the total complexity

of the algorithm effectively since **Step 2** is only responsible for a portion of algorithm complexity.

Algorithm 3 New Search Algorithm to Form $\mathbf{A_{IF}}$.

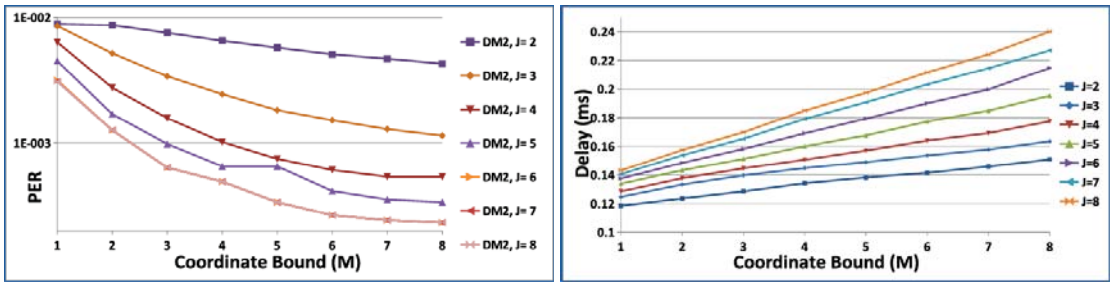
Input: $\mathbf{Q}$, Ω.
Output: Full rank matrix $\mathbf{A_{IF}}$.
Step 1: Let $f(\mathbf{t}) = \mathbf{t^T Q t}$ and sort all vectors in the searching candidate set Ω such that:

$$\Omega = \{\mathbf{t_1}, ..., \mathbf{t}_{|\Omega|} : f(\mathbf{t}_1) \leqslant ... \leqslant f(\mathbf{t}_{|\Omega|})\}$$

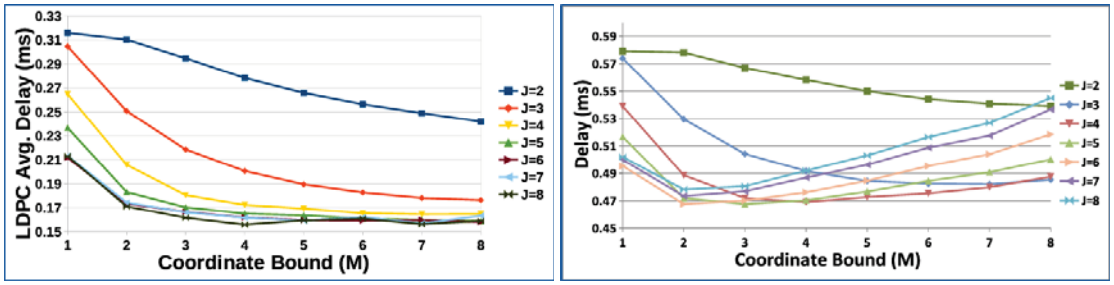
and set the $\mathbf{a_1} = \mathbf{t_1}$, initialize $i = 1$, and $j = 1$.
Step 2: Form $\mathbf{G}$ from the first L vectors of Ω, and calculate $\mathbf{R}$=qr($\mathbf{G}$), $i = 8$, j=sum(diag(abs($\mathbf{R}$) > 0))
While $i < |\Omega|$ and $j < L$, do
(i) Remove $L - j$ vectors from the $\mathbf{G}$ that diag(abs($\mathbf{R}$))=0
(() ii) Append the $L - j$ next vectors of Ω to $\mathbf{G}$
(iii) $i = i + L - j$, $\mathbf{R}$=qr($\mathbf{G}$), j=sum(diag(abs($\mathbf{R}$) > 0))
Step 3: If $j = L$, then $\mathbf{A_{IF}} = \{\mathbf{a_1}, ..., \mathbf{a_L}\}$, else $\mathbf{A_{IF}} = \mathbf{A_{MMSE}}$.

3.4 Evaluation and Results

To evaluate the error rate performance and time complexity we performed a case study.The case study is based on the IEEE 802.16-2012 standard, 4×4 MIMO antenna arrangement, with 16 QAM modulation. The system is using rate $\frac{1}{2}$ LDPC coding with packet size 2304 packet size. The channel matrix is considered to be constant within a packet transmission (slow fading). The LDPC decoding is iterative and a maximum of 100 iteration is used. The decoder checks the parity checks in each iteration and if the checks are met the LDPC decoder iterations are stopped. The LDPC decoder uses log likelihood ratio method for variable node update. These settings are known to perform close to optimum performance [3].



(a) DM2 performance for dif. J and M (b) $\mathbf{B_{IF}}$ computation time for dif. J and M



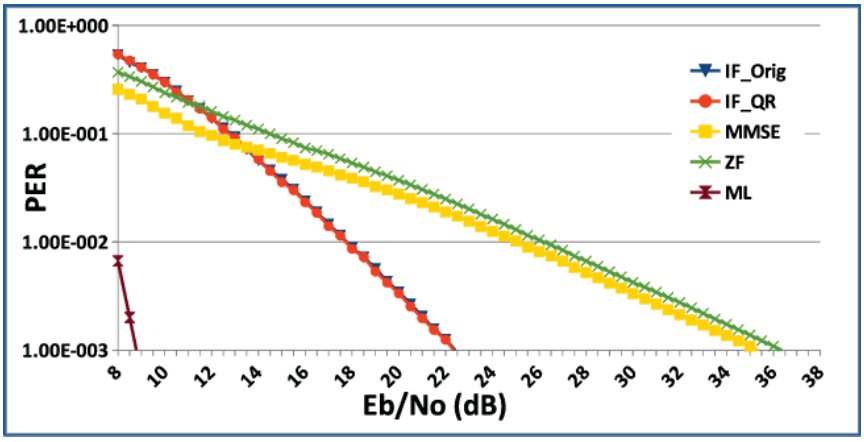
(c) LDPC computation time for dif. J and M (d) DM2 computation time for dif. J and M

Figure 3.2: Performance curves (based on PER) and computation times of DM1 and DM2 for various *J, M*. Eb/No=25 dB

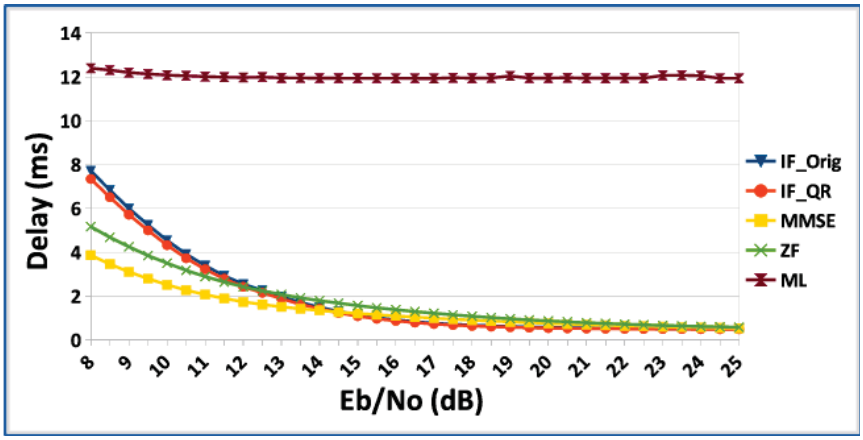
Fig. 3.2a shows the performance of DM2 for arbitrary Eb/No of 25[16] dB with different J and M. The performance of DM2 is almost optimum for J and M larger than six. Although for J and M larger than six the performance is almost optimum, slight improvement is achievable with increasing the M boundary to eight. It is worth mentioning again that these numbers are different with results of [4] since authors in [4] consider the probability of successive construction of $\mathbf{B_{IF}}$ while our target is optimization of PER performance curve.

Increase in the J and M bound will increase the time complexity linearly; this can be seen in Fig. 3.2b. The information of the hardware and software used for different algorithms are as follow:

[16]Where PER of MMSE detector is 10^{-3} and ideal candidate for Performance comparison with DM2



(a) Different detectors performance vs. Eb/No



(b) Different detectors delays vs. Eb/No

Figure 3.3: Performance and computation times of different detectors Vs. Eb/No

- Intel(R) Core(TM) i7 CPU 920 @ 2.67 GHz.
- 9 GB RAM.
- 64 bit Windows operating system.
- Simulink (Version 8.2)
- Parallel processing is not enabled.

We chose *J=6* and *M=8* since it has the minimum computation time for construction of $\mathbf{B_{IF}}$ while it provides the optimum performance.

In Fig. 3.2c the LDPC decoder computation time for different *J* and *M* presented. The higher computation time for LDPC means higher average number of iterations. Since in general with very low values of *J* and *M* the probability of successful construction of $\mathbf{B_{IF}}$ is lower, a higher decoding time is required. As the result the LDPC computation time is higher with lower *J* and *M*.

Fig. 3.2d shows the receiver computation time (receiver computation time = detector computation time + decoder computation time) of DM2 for different *J* and *M*. All the curves except *J=2* are concave (when *J=2*, the LDPC computation time is dominant for all *M* coordinates.) The concave curve is the result of effective decrease in decoder (LDPC) computation time with the increase in *J* and *M* limits; while with further increase in these parameters the computation time of detector increases and it is dominant. It worth mentioning that there is a high correlation (covariance of 0.98) between standard deviation (SD) of receiver computation time and average LDPC iterations.

The performance of different detectors are presented in Fig. 3.3a. The simulations are done for 10^8 packets or 1000 packet errors which ever happens first. The *J=6, M=8* is considered for DM2 in this simulations. The soft detector uses approximate log likelihood ratio (LLR). The ML detector uses depth first search list decoding algorithm (DFS-LSD) with list size of 32 and LLR clipping of eight. Simulation results show that while the MMSE performs 1 dB better than ZF detector at PER of 1%, DM2 provides 8.5 dB better performance. This reduces the gab between MMSE and ML detector by 45%.

Figure 3.3b shows the average computation time of different receiver in various Eb/No. The difference in computation times for each detector in different Eb/No is mostly depends on the number of LDPC iterations that is required. Each point in the curve is the result of average over 10^5 packets. With the increase in channel Eb/No the receiver computation time will decrease due to the decrease in decoder computation time. This is more notable in linear detectors since the computation time of the detector is considerably lower.

Table 3.2: Different detectors computation time.

Detector	**ZF**	**MMSE**	$\mathbf{B_{IF\_Orig}}$	**IF_Orig**	$\mathbf{B_{IF\_QR}}$	**IF_QR**	**ML**
Time (ms)	0.122	0.121	0.233	0.378	0.222	0.360	11.84

The detectors computation time is constant and presented in Table 3.2. The result shows that the time complexity of ZF[17] is 1% higher than MMSE, while the PER performance of MMSE detector is 1 dB better and the MMSE and ZF have the same slope performance

[17]The time complexity of calculating the inverse matrix is lower in MMSE detector.

curves (with the increase in Eb/No of the channel the performance difference is constant). There is a 13 dB difference between MMSE and DM2 at PER of 10^{-3} with 2.96× increase in computation time; while the computation time of ML detector is 32.87× higher compared to DM2 and provides 13 dB better performance. The gap between MMSE and ZF increases with increase in Eb/No; in another word, their performance difference increases targeting lower error rates. The notable result is that the IF detector reduces the performance gap between ZF and ML detector by 52% (at PER of 10^{-3}) with only 2.96× increase in time complexity, while the ML detector is 97.2× more complex in time.

To increase the throughput based on the CPU architecture utilizing upto eight parallel detection unit (parallel path) is considered. Each data packet is delivered to one detector and detectors work independently. Table 3.3 reports the achievable uncoded data throughput for each detection method using the provided hardware. Each row of the table shows the results achieved utilizing different number of parallel detection units. Using eight parallel path the DM2 troughpu increases by 5.1× while this is 5.85× for ML detector.

Table 3.3: Achievable data throughput (Mbps) for each detection method utilizing different number of parallel paths.

# of Parallel Paths	ZF	MMSE	IF_QR	IF_Orig	ML
1	18.92	19.12	6.74	6.42	1.94
2	37.34	37.95	13.33	12.7	3.64
3	49.49	49.42	18.75	17.86	5.33
4	58.71	61.15	23.27	22.17	6.64
5	67.31	68.36	26.8	25.53	7.86
6	74.78	77.92	28.79	27.43	8.99
7	88.96	84.11	31.43	29.95	10.33
8	89.33	90.39	34.4	32.78	11.37

Table 3.4: Average number of FLOPs required for calculation of One transmit Ppacket in different arrangements.

Arrangment	ZF	MMSE	IF_Orig	IF_QR	ML
4×4 **16 QAM**	38812	38908	177938	173362	41610285
4×4 **64 QAM**	30364	30436	161042	156466	207131142
8×8 **16 QAM**	84352	84672	671994	559780	587898898

3.4.1 Complexity and Higher Orders

In this section we compare the complexity of different detection methods using the traditional method (number of FLOPs required to complete the tasks), and also measuring the average time complexity required by the simulation platform to complete the tasks. In this comparison our base setup is a 4×4 MIMO channel with 16 QAM modulation, and to demonstrate the effect of increasing the modulation constellations and number of antenna we present the same measurements on two other setups: 8×8 antenna arrangement with 16 QAM modulation, as well as 4×4 antenna arrangement with modulation of 64 QAM.

Table 3.4 presents the average number of FLOPs required for calculation of one transmit packet in different modulations and antenna arrangements. Even though, the FLOPS numbers are good indicators of the complexity of a task in recent general purpose architectures due to advanced techniques (out of order processing, single instruction multiple data (SIMD), multiple instructions multiple data, and specially parallel processing and presence of multi-core CPUs) utilized, the number of FLOPs is not a good indicator of how long it takes for an special setup to finish a task.

In case of modern general purpose architectures dependency between the code lines is

Table 3.5: Average required time complexity for calculation of one packet in different arrangements (ms).

Arrangment	ZF	MMSE	IF_Orig	IF_QR	ML
4×4 **16 QAM**	0.122	0.121	0.378	0.360	11.84
4×4 **64 QAM**	0.125	0.124	0.38	0.362	67.04
8×8 **16 QAM**	0.154	0.152	1.18	1.345	144

the key factor in determining the required time complexity. Table 3.5 shows the average required time complexity for mentioned setup to calculate one packet using different detection methods. The FLOPs count and time complexity numbers show same pattern when the detection method is kept same; however, the comparison between different detection methods does not hold the same pattern. Also, it worth to mention that for linear detectors even though the Flops count reduces with increase in modulation orders, the time complexity of linear detectors shows slight increase. This is related to the fact that in higher modulation order the number of symbols in a packet is less and that in fact reduces the parallel processing ability of the task.

3.4.2 Presicion Analysis

An important information when analyzing the complexity of a system is to know what is the bit precision reacquired for completing each task. This is in particular important when a fixed point processor is assigned to complete the task our when hardware implementation of an algorithm is targeted. This is also important in general purpose floating point processors when SIMD is utilized.

We did the fixed point precision analysis with the goal of finding minimum precision

required so that the maximum change in performance curve within the SNR range of 8 dB to 23 dB is less than 0.05 dB. Some of the key elements of such an implementation is mentioned here after.

The signed two's complement is used trough the entire system. No protection on saturation is utilized and the word length of binary representation is determined using adaptive scaling method with now protection on overflow except for final calculated LLRs which are limited to [-2,2) range. The truncation is used for list significant bit (LSB) precision handling. We use the following notation for presenting the floating point analysis results. XsfixY shows a signed fixed point representation, where X is is the total number of word length in bits including the sign bit and Y is the number of precision bits. This representation assumes that the real world values remain same.

Our simulations shows that at least 13sfix10 and 14sfix9 is required for quantization of the channel matrix and received data accordingly. The channel required precision remains the same for matrix **Q**. The required Precision for calculating the Eigen values and Eigen vectors is 9sfix8. The output of the search algorithm needs 5sfix0 for $\mathbf{A_{IF}}$ and 14sfix9 for $\mathbf{B_{IF}}$. The maximum precision is required by matrix inverse operation which is 33sfix22 and LLRs need 7sfix5 to provide close to optimum performance.

3.5 Conclusion

We studied the possibility of a SDR solution for MIMO system using IF receiver on a general propose Intel i7 processor. We introduce two low complexity soft demapping methods. Using these two method we were able to acquire the benefits of Integer Forcing receiver while maintaining the gain achieved from low complexity soft demapping for QAM modulation. The IF detector reduces the gap between ZF and ML detectors by 52% with only 2.96$\times$ more complexity while LSD-DFS detector has 97.2$\times$ higher time complexity compared to ZF detector. The total time required by IF detector for detection of one packet of Data is 0.36 ms with the target hardware. While the achieved throughput is lower than the requirements of high throughput data communication systems (5G), this results can be used as comparison source for more powerful hardware/software setup of SDR. While the complexity of a ML detector is high for a software radio and the performance of a linear detector is sub optimum; the IF detector provides a judicious compromise between two solutions. While the complexity of a ML detector is high for a software radio and the performance of a linear detector is sub optimum; the IF detector provides a judicious compromise between two solutions.

4 THE NORMALIZED SINGULAR VALUE DECOMPOSITION OF NON-SYMMETRIC MATRICES USING GIVENS FAST ROTATIONS

In this work we introduce the algorithm and the fixed point hardware to calculate the normalized singular value decomposition of a non-symmetric matrices using Givens fast (approximate) rotations. This algorithm only uses the basic combinational logic modules such as adders, multiplexers, encoders, Barrel shifters (B-shifters), and comparators and does not use any lookup table. This method in fact combines the iterative properties of singular value decomposition method and CORDIC method in one single iteration. The introduced architecture is a systolic architecture that uses two different types of processors, diagonal and non-diagonal processors. The diagonal processor calculates, transmits and applies the horizontal and vertical rotations, while the non-diagonal processor uses a fully combinational architecture to receive, and apply the rotations. The diagonal processor uses priority encoders, Barrel shifters, and comparators to calculate the rotation angles. Both processors use a series of adders to apply the rotation angles. The design presented in this work provides $2.83 \sim 649$ times better energy per matrix performance compared to the state of the art designs. This performance achieved without the employment of pipelining; a better performance advantage is expected to be achieved employing pipelining.

4.1 Introduction

Emerging technologies require a high performance, low power, and efficient solution for singular value decomposition (SVD) of matrices. A High performance and throughput hardware implementation of SVD is necessary in applications such as linear receivers for 5G MIMO telecommunication systems [4], various real-time applications [36], classification in genomic signal processing [37], and learning algorithm in active deep learning [38]. Matrix decomposition, and more specifically, SVD is also the most commonly used DSP algorithm and often is the bottle neck of various computationally intensive algorithms. For instance this is paramount for some the resent studies including performance analysis of adaptive MIMO transmission in a cellular system [39], image compression [40], and new image processing techniques of face recognition [41].

The design of SVD arithmetic unit has been vastly investigated by the researchers. For effective implementation of SVD in the hardware, [5] presents BLV algorithm with a systolic architecture. This architecture uses a set of diagonal processors (DP) and a set of non-diagonal processors (NDP). The DP processor calculates, applies and transmits the horizontal (θ_H) and vertical (θ_V) rotation angles while NDP applies the received rotation angles. To calculate the division, square root, and multiplication required for this method Cavallaro uses the coordinate rotation digital computer algorithm (CORDIC) [42]. CORDIC algorithm is mainly credited to Jack Volder. This algorithm was originally introduced for solving the problem of real-time navigation [43]. CORDIC algorithm has

been used in different math coprocessor [44], digital signal processors [45], and software defined radios [46]. Different implementation of CORDIC algorithm have been introduced, including but not limited to Higher Radix CORDIC algorithms [47], Angle Recoding methods [48], Hybrid and Coarse-Fine Rotation CORDIC [49], Redundant-Number-Based CORDIC implementation [50], Pipelined CORDIC architecture [51], and Differential CORDIC algorithm [52]. A relatively comprehensive review of CORDIC algorithm is presented in [53]. To reduce the implementation complexity of SVD, different optimization have been proposed. Delsome proposed double rotations to avoid square roots and divisions for scaling [54].

In 1991 Gotze introduces an algorithm that combines the inner-iterations (CORDIC iterations) with outer-iterations of BLV algorithm (sweep). This method instead of calculating the accurate rotations, calculates the fast rotation (Givens fast rotations also known as Givens approximate rotations) angles which is equivalent to one iteration of CORDIC algorithm [7], [6]. As the result this hardware does not require any look-up table, and calculating the rotation angles requires only one clock cycle. However this method has the following disadvantages:

1. The hardware implementation requires floating-point arithmetic.
2. The algorithm only works for symmetric matrices.
3. Using this method the BLV algorithm loses its quadratic convergence speed (quadratic to number of sweeps).
4. The proposed algorithm does not provide the "Normalized" results.

In this work we address the disadvantages of Givens fast rotations. The proposed hardware does not require the floating-point arithmetic, and as the result, it does not need the pre-processing (alignment of exponents) and post-processing (renormalization of matrices) blocks mentioned in implementations of Givens fast rotation. The algorithm that we proposed is able to handle symmetric as well as non-symmetric matrices (calculating the rotations for non-symmetric matrices require the calculation of two intermediate variable angles, proximate calculation of these two intermediate variables makes the calculation of rotation angles challenging and we were able to offer an adaptive solution for it). Also, the proposed algorithm provides the "Normalized" results. The rest of this work is organized as follows: First we introduce the method to merge the NSVD algorithm [5] with Givens fast rotations and Delsome double rotations method [55]. The result (ERNSVD algorithm) is similar to the Gotze work in [6] to find the Eigenvalues of a symetric matrix exept the algorithm is able to calculate the decomposition of a non-symetric matrix. In addition we introduce a method that directly calculates the horizontal and vertical fast rotations using the Forsythe and Henrici SVD (FHSVD) algorithm and called it expedite rotations SVD (ERFHSVD) [5]. We present a possible hardware implementation and provisions that make the fixed point implementation of these algorithm possible in section 4.3. The hardware implementation is a design for decomposing a 2×2 matrix, as the basic building block for decomposition of matrices with larger size. We present and compare the implementation result of our design with some of the state of the art designs in section 5.5. Finally we conclude this Work in section 5.6.

4.2 Algorithm of Calculating Fast Rotation Angles for Non-Symmetric Matrices

This work is inspired by the Normalized SVD (NSVD) and FHSVD algorithm presented in [5]. We use the Givens fast rotations presented in [6] and double rotation by Delsome [54] to reduce the implementation complexity. In this section we briefly review the bases of our inspiration (NSVD, FHSVD algorithms, double rotations, and fast rotations) and introduce our measures (approximation error and norm of off-diagonal elements) for evaluation and comparison of different methods through this work. This section holds four subsections containing our three proposed algorithms as well as their comparison and also study of the relaxing the boundary conditions which will result in reduction in hardware complexity.

The fast rotation algorithm tries to find the closest angle ($\hat{x}$) to any rotation angle (x) such that: first, $|\hat{x}| \leq \frac{\pi}{4}$ and also $|tan(\hat{x})| = 2^{-l}; \quad l \geq 0$. The fast rotations do not symmetrize or diagonalize the matrix in one rotation, in fact they generate a more symmetric matrix with each rotation or they reduce the off-diagonal norm of the matrix; using this method the quadratic convergence properties of SVD algorithm is lost. We use the Delsome proposed double rotations method which for any rotation angle uses $\bar{\gamma} = \frac{\gamma}{2}$ as rotation angles and applies the rotations twice [55]. This technique will eliminate the need for the calculation of square root function and division for scaling. This means adding one to l and applying two rotations with the angle equal to $arctan(2^{-l+1})$. We use $\tilde{}$ for noting any angle or its tangent when Delsome double rotation and Gotze approximation are applied.

We assume the 2×2 matrix A and the decomposed matrix is defined as in (5.10) for the review of NSVD and FHSVD algorithms.

$$\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \mathbf{U} \times \mathbf{\Sigma} \times \mathbf{V}^T \tag{4.1}$$

The NSVD algorithm calculates the first rotation to symmetrize the matrix using (4.2, and then using (4.3) generates the symmetric matrix. NSVD then uses (4.4) to find the diagonalizing rotations.

$$\rho = tan^{-1}(\frac{c+b}{d-a}) \tag{4.2}$$

$$\mathbf{B} = \mathbf{R}_\rho \times \mathbf{A} = \begin{bmatrix} cos(\rho) & sin(\rho) \\ -sin(\rho) & cos(\rho) \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} p & q \\ q & r \end{bmatrix} \tag{4.3}$$

$$\phi = tan^{-1}(\frac{2q}{q-p}) \tag{4.4}$$

$$\begin{aligned} \mathbf{\Sigma} = \mathbf{R}_\phi^T \times \mathbf{B} \times \mathbf{R}_\phi = \begin{bmatrix} d_1 & 0 \\ 0 & d_2 \end{bmatrix} \\ = \begin{bmatrix} cos(\phi) & sin(\phi) \\ -sin(\phi) & cos(\phi) \end{bmatrix}^T \begin{bmatrix} p & q \\ q & r \end{bmatrix} \begin{bmatrix} cos(\phi) & sin(\phi) \\ -sin(\phi) & cos(\phi) \end{bmatrix} \end{aligned} \tag{4.5}$$

The FHSVD algorithm first calculates the α and β using (5.11) and (5.12) as the intermediate values, and using (5.13) and (5.14) the algorithm calculates the horizontal (Θ) and vertical (θ) rotations. In the last step FHSVD algorithm swaps the values of *sin* and *cos* and change the sign of these values if needed to make sure $d_1 \geq d_2$.

$$\alpha = tan^{-1}(\frac{c+b}{d-a}) \tag{4.6}$$

$$\beta = tan^{-1}(\frac{c-b}{d+a}) \tag{4.7}$$

$$\Theta = \frac{\alpha+\beta}{2} \tag{4.8}$$

$$\theta = \frac{\alpha-\beta}{2} \tag{4.9}$$

$$\begin{aligned} \mathbf{\Sigma} = \mathbf{R}_\theta^T \times \mathbf{A} \times \mathbf{R}_\Theta = \begin{bmatrix} d_1 & 0 \\ 0 & d_2 \end{bmatrix} \\ = \begin{bmatrix} cos(\theta) & sin(\theta) \\ -sin(\theta) & cos(\theta) \end{bmatrix}^T \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} cos(\Theta) & sin(\Theta) \\ -sin(\Theta) & cos(\Theta) \end{bmatrix} \end{aligned} \tag{4.10}$$

[6] offers a floating point implementation and uses the exponent bits to calculate l. In fix point representation we introduce (4.12) to replace the exponent bits value of the original algorithm. This will reduce the function $exp_2(x)$ to a priority encoder applied on x, assuming x is an integer number. We will discuss this in more details in 4.3. Equation system (4.11) is equal to checking the most significant bit (MSB) in two's-complement representation.

$$Sign(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases} \tag{4.11}$$

$$(exp_2(x), v) = \begin{cases} (0,0) & \text{if } x = 0 \\ (\lfloor log_2{}^{|x|} \rfloor, 1) & \text{if } x \neq 0 \end{cases} \tag{4.12}$$

There are two measures that are normally used in different Jacobi based decomposition.

1. The approximation error $\mid d \mid$ in [6] for a 2×2 symmetric matrix defined as the

absolute value of off diagonal element before and after the application of k^{th} rotation (4.13). The smaller $\mid d \mid$ shows a more accurate approximation.

$$\mid d \mid = \mid \frac{q_{(k+1)}}{q_k} \mid \tag{4.13}$$

This is applied to the 2×2 matrix, and $\mid d \mid_{Max} < 1$ is called error bound and is one of the two conditions to assure the convergence of the algorithm. The other condition is to keep the orthogonality of the rotation matrix. We plan to apply this measures to non-symmetric matrices so we extend the equation (4.13) to $\mid\mid D \mid\mid$. Note that $\mid d \mid$ is the same as $\mid\mid D \mid\mid$ if the matrix is symmetric.

$$\mid\mid D \mid\mid = \frac{\sqrt{b_{k+1}^2 + c_{k+1}^2}}{\sqrt{b_k^2 + c_k^2}} \tag{4.14}$$

The approximation error (as the dependent variable) is normally measured for different values of τ (as the independent variable) as defined in (4.15).

$$\tau = \frac{r-p}{2q} \tag{4.15}$$

To extend equation (4.15) to an independent variable applicable to non-symmetric matrices, we define τ_1 and τ_2 in (4.16) and (4.17) accordingly. $\frac{1}{\tau_2} = 0$ and $\tau_1 = \tau$ if the matrix is symmetric.

$$\tau_1 = \frac{d-a}{b+c} \tag{4.16}$$

$$\tau_2 = \frac{d+a}{b-c} \tag{4.17}$$

2. The Norm of the off-diagonal elements of a matrix versus (Vs.) the number of sweeps is the second metric used for measuring the quality of any fast rotation methods as well as the comparison of the diagonalization speed in different methods. In [6]

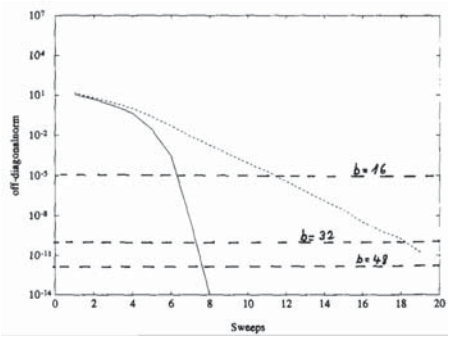


Figure 4.1: Off-diagonal Norm vs. number of sweeps [6]

this value is calculated for a 70×70 matrix. The elements of matrix are randomly generated numbers of normal distribution. We run the same test for 100 times and calculate the RMS (root mean square) of the off-diagonal norms (RMS_{ODN}) to keep the comparability and also keep the results accurate.

Figure 4.1 shows the Off-diagonal Norm of a 70×70 vs. number of sweeps for the fast rotations (dotted curve) and the original Givens rotations (solid line). The dashed lines show the accuracy achievable by different number of bits. This figure provided in [6] is to support the fast rotations method with the following explanation: While achieving the accuracy of 16 bits or better; the original method requires seven iterations versus twelve iterations in the fast rotation method. One must realize that the calculation of 16-bit exact sine and cosine values requires more complexity. As an example if the sine and cosine are implemented using CORDIC method, it requires 16 inner iteration to calculate these values. In the next two subsection we will present two algorithms that are basic blocks of eminent rotations NSVD (ERNSVD).

4.2.1 Symmetrizing Algorithm

Algorithm 4 is the approximation to the rotation angle achieved from equation (4.2). Boundaries in Step 2 of the algorithm are based on the suggestion in [7] that guaranties $|d| \leq \frac{1}{3}$. Authors in [6] explain that using Delsome double rotations this limit does not hold any more (this method will guaranties $|d| \leq \frac{7}{12}$); however, the authors explain that this is not an issue since the approximation error merges to the original bounds when the rotation angles get smaller. It should be mentioned that the bounds to approximation error is still less than one.

Algorithm 4 Algorithm to Calculate the Symmetrizing Fast Rotations for Non-Symmetric Matrices.

Input: **A**.
Output: Rotation Matrix $\mathbf{R}_{\tilde{\rho}}$.
Step 1: Calculate the initial values:

$$S_N = Sign(b-c)$$
$$S_D = Sign(d+a)$$
$$N = |b-c|$$
$$D = |d+a|$$
$$K = exp_2(D) - exp_2(N)$$

Step 2: Calculate $l_{\tilde{\rho}}$ using following case statement:

$$l_{\tilde{\rho}} = \begin{cases} K+1 & \text{if } 1.5 \times D > (2^{K+1} - 2^{-K}) \times N \\ K-1 & \text{if } 1.5 \times D < (2^{K} - 2^{-K+1}) \times N \\ K & \text{default} \end{cases}$$

$$l_{\tilde{\rho}} = max(l_{\tilde{\rho}} + 1, 1)$$

Step 3: Calculate the (c,s) pair using the following case statement:

$$\tilde{t} = 2^{-l_{\tilde{\rho}}}$$

$$(c,s) = \begin{cases} (1,0) & \text{if } N = 0 \\ (0,1) & \text{if } D = 0 \\ \frac{1}{1+\tilde{t}^2} \times (1-\tilde{t}^2, 2 \times S_N \times S_D \times \tilde{t}) & \text{default} \end{cases}$$

Step 4: Calculate the Symmetrizing $\mathbf{R}_{\tilde{\rho}}$:

$$\mathbf{R}_{\tilde{\rho}} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}$$

4.2.2 Diagonalizing Algorithm

Algorithm 5 is the approximation to the rotation angle achieved from equation (4.4). Figure 4.2 demonstrates the $|| D ||$ vs. τ when Algorithm 5 is applied on a symmetric matrix. Figure 4.3 demonstrates the $|| D ||$ vs. τ_1 and τ_2 when Algorithm 5 is applied on a non-symmetric matrix with ideal symmetrizing rotations. The fast rotations NSVD (FRNSVD) algorithm is based on applying the fast symmetrizing and fast diagonalizing algorithms on a given matrix.

Figure 4.4 demonstrates the $|| D ||$ vs. τ_1 and τ_2 when Algorithm 4 and Algorithm 5 is applied on a non-symmetric matrix. The approximation errors are typically higher in this method compared to Figure 4.3. This increase in the approximation error is expected since the approximation error of two algorithms can boost the total approximation error. Figure 4.6 compares the RMS_{ODN} for NSVD, FRNSVD, and ERNSVD. The error floor that happens in iterations 19th and after in ERNSVD, and FRNSVD are due to the fixed point (32 bit) implementation of the algorithm.

4.2.3 Relaxing the Boundary Conditions

Implementing the case statement in **Step 2** of both algorithms beside the two comparators requires two Barrel shifters,two adders to calculate the coefficients of N and an adder to calculate $1.5 \times D$. We can reduce the complexity of conditions by using (4.18). This reduces the complexity of case statement to two comparators, an adder, and a Barrel shifter.

Algorithm 5 Algorithm to Calculate the fast Diagonalizing Rotations for Symmetric Matrices.

Input: **B**.
Output: Rotation Matrix $\mathbf{R}_{\tilde{\phi}}$.
Step 1: Calculate the initial values:

$$S_N = Sign(b+c)$$
$$S_D = Sign(d-a)$$
$$N = |c+b|$$
$$D = |d-a|$$
$$K = exp_2(D) - exp_2(N)$$

Step 2: Calculate $l_{\tilde{\phi}}$ using following case statement:

$$l_{\hat{\phi}} = \begin{cases} K+1 & \text{if } 1.5 \times D > (2^{K+1} - 2^{-K}) \times N \\ K-1 & \text{if } 1.5 \times D < (2^{K} - 2^{-K+1}) \times N \\ K & \text{default} \end{cases}$$

$$l_{\tilde{\phi}} = max(l_{\phi} + 2, 1)$$

Step 3: Calculate the (c,s) pair using the following case statement:

$$\tilde{t} = 2^{-l_{\tilde{\phi}}}$$

$$(c,s) = \begin{cases} (1,0) & \text{if } N = 0 \\ (0,S_N) & \text{if } D = 0 \\ \frac{1}{1+\tilde{t}^2} \times (1-\tilde{t}^2, 2 \times S_N \times S_D \times \tilde{t}) & \text{default} \end{cases}$$

Step 4: Calculate the diagonalizing $\mathbf{R}_{\tilde{\phi}}$ using the following case statement:

$$\mathbf{R}_{\tilde{\phi}} = \begin{cases} \begin{bmatrix} c & s \\ -s & c \end{bmatrix} & \text{if } S_N < 0 \\ \\ \begin{bmatrix} s & c \\ c & -s \end{bmatrix} & \text{else} \end{cases}$$

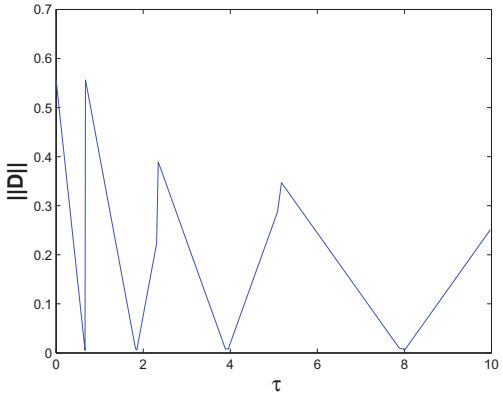


Figure 4.2: $||\ D\ ||$ vs. τ for symmetric matrix when Algorithm 5 is applied.

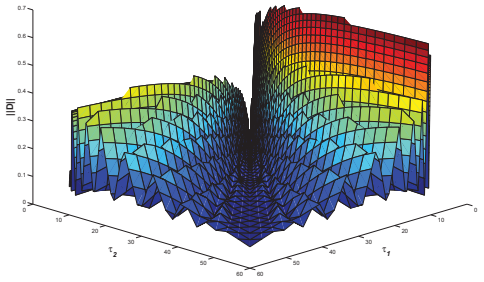


Figure 4.3: $||\, D \,||$ vs. τ_1 and τ_2 when Algorithm 5 is applied for non-symmetric matrix with ideal symmetrization.

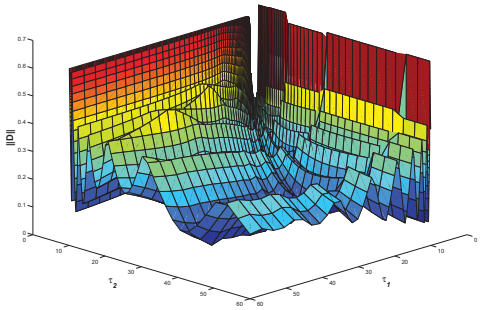


Figure 4.4: $||\, D \,||$ vs. τ_1 and τ_2 for non-symmetric matrix for FRNSVD.

We study the effect of relaxing the conditions using both of the measures. Equation (4.18) provides an approximation that is accurate for higher ls while it is inaccurate for small ls. The solution is in limiting the rotation angles to smaller rotations (changing $l = max(l+1,1)$ to $l = max(l+1,2)$); in another word, the rotation angles are over estimated when the original rotation is closer to $\frac{\pi}{4}$. The empirical results of the simulation shows that changing both l_ρ and l_ϕ will result in reduced convergence speed. We found that the best combination is achieved by limiting the l_ρ to minimum of two while l_ϕ can get any positive integer value.

$$l = \begin{cases} K+1 & \text{if } 1.5 \times D > 2^{K+1} \times N \\ K-1 & \text{if } 1.5 \times D < 2^{K} \times N \\ K & \text{default} \end{cases} \tag{4.18}$$

We called the relaxed version of fast rotations the expedite rotations NSVD (ERNSVD). Figure 4.6 compares the RMS_{ODN} for NSVD, FRNSVD and ERNSVD. The RMS_{ODN} are very close for both methods and with ERNSVD being less complex the new boundaries are studied after this point. Figure 4.5 demonstrates the $||\ D\ ||$ vs. τ_1 and τ_2 for ERNSVD. The FRNSVD has lower approximation error compare to ERNSVD when τ_1 or τ_2 are closer to zero. This is the effect of the empirical change we applied to the boundaries in equation (4.18).

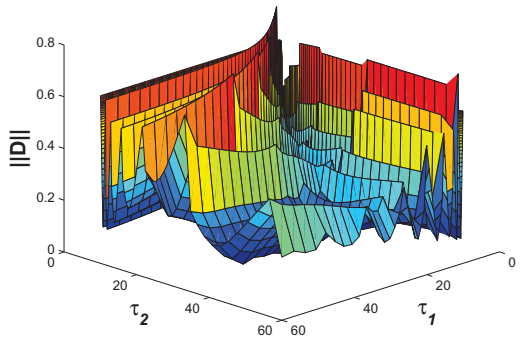


Figure 4.5: $||\ D\ ||$ vs. τ_1 and τ_2 for non-symmetric matrix for ERNSVD.

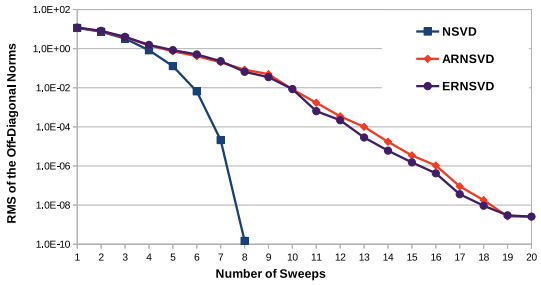


Figure 4.6: Comparison of the RMS_{ODN} for NSVD, FRNSVD, ERNSVD algorithms.

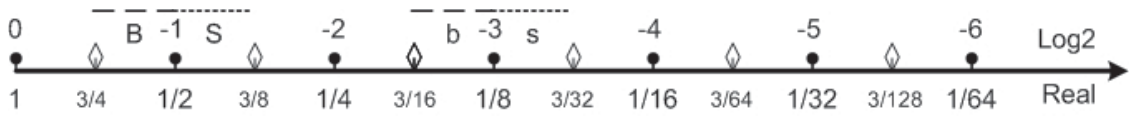


Figure 4.7: Axis showing the boundary conditions of each fast rotation angle for relaxed version

4.2.4 Direct Estimate Algorithm

One major contribution of this work is the algorithm that enable us to directly calculate the fast rotations for non-symmetric matrices based on FHSVD algorithm. Assuming a 2×2 matrix is defined as in equation (5.10), FHSVD algorithm uses (5.11) and (5.12) to calculate α and β; then using those two values and equation (5.13) and (5.14), it calculates the vertical and horizontal Givens rotations. Note that for traditional rotation algorithm (proposed by Gotze [6]), the value of β is zero, and we only need to calculate the value of α (Gotze algorithm is for symmetric matrices only).

While Gotze method [6] is using the same flow as in the calculation of accurate rotation angles and then divides the results by two to take benefit of double rotations, we use $tan(x) \approx x$ and the fact that this approximation is more accurate for smaller angels ($lim_{x \longrightarrow 0}^{tan(x)} = x$), to provide more accurate estimation of rotation angels.

Figure 4.7 shows the boundaries for the relaxed version with a rhombus indicator on the axis while the approximated angles are shown with circle indicators. This shows that as an example if the range of the tangent is between $(\frac{3}{8}, \frac{3}{4})$ its approximation will be $\frac{1}{2}$. The notion **B** or **b** are used when the original rotation angle is larger than approximations while **B** is used for the bigger angle between α and β. The notion **S** or **s** are used when the rotation angle is smaller than its approximation while **L** is used for the bigger angle between α and β. Knowing if the fast rotation angle is overestimating or underestimating the original rotation angle might requires an extra comparator (depends on the implementation) in

angle calculation circuit but it could provide considerable benefit as we will demonstrate in following sections

TABLE 4.1 gives a better understanding of how the rotation angles ($\tilde{\theta}$ and $\tilde{\Theta}$) are decided only based on having an approximation of α and β. $\tilde{\alpha}$ and $\tilde{\beta}$ are approximations to α and β that Delsome double rotation and angle approximation are both applied (In TABLE 4.1 α and β are half the rotation angles achieved from (5.11) and (5.12) to take advantage of tangent properties: $lim_{x\longrightarrow 0}^{tan(x)} = x$). Column θ in the table shows the possible range of θ based on the value of $\tilde{\alpha}$, $\tilde{\beta}$, and if the approximation angels are larger or smaller than the original rotation angles. $\tilde{\theta}_{big}$ is the approximation if we use the biggest possible rotation angle in the range, and $\tilde{\theta}_{small}$ is the approximation if we use the smallest possible rotation angle in the range (same notation is applied to $\tilde{\Theta}_{big}$ and $\tilde{\Theta}_{small}$). Choosing the bigger rotation angles in general might result in increased floor level of RMS_{ODN} while it results in a faster convergence rate. Columns $\tilde{\theta}$ and $\tilde{\Theta}$ in the table show the approximation we used in algorithm 7 as an example; however, this does not mean that any application of the algorithm has to use the same numbers. In fact we urge a search on the possibilities based on the application.

4.2.5 Reducing the Direct Estimation Complexity

Algorithm 7 shows the complete flow for the calculation of $\tilde{\theta}$ and $\tilde{\Theta}$. The complexity of angle calculation is optimized for achieving reasonable hardware complexity. In **Step2** the calculation of $(l1_{temp}, B)$ or $(l2_{temp}, b)$ can be reduced to two comparison if we do not

Algorithm 6 Algorithm to Directly Calculate the fast Rotation Angles for Non-Symmetric Matrices.

Input: **A**.
Output: Rotation Matrix $\mathbf{R}_{\tilde{\theta}}$ and $\mathbf{R}_{\tilde{\Theta}}$.
Step 1: Calculate the initial values:

$$S_{N1} = Sign(c+b)$$
$$S_{D1} = Sign(d-a)$$
$$N1 = |c+b|$$
$$D1 = 2\times|d-a|$$
$$K1 = exp_2(D1) - exp_2(N1)$$
$$S_{N2} = Sign(c-b)$$
$$S_{D2} = Sign(d+a)$$
$$N2 = |c+b|$$
$$D2 = 2\times|d-a|$$
$$K2 = exp_2(D2) - exp_2(N2)$$

Step 2: Calculate l_α and l_β using following case statement:

$$(l1_{temp}, B) = \begin{cases} (K1+1,1) & \text{if } 1.5\times D1 > 2^{K1+1}\times N1 \\ (K1-1,0) & \text{if } 1.5\times D1 < 2^{K1}\times N1 \\ (K1,1) & \text{if } D1 < 2^{K1}\times N1 \\ (K1,0) & \text{default} \end{cases}$$

$$l_\alpha = max(l1_{temp}+1,2)$$

$$(l2_{temp}, b) = \begin{cases} (K2+1,1) & \text{if } 1.5\times D2 > 2^{K2+1}\times N1 \\ (K2-1,0) & \text{if } 1.5\times D2 < 2^{K2}\times N1 \\ (K2,1) & \text{if } D2 < 2^{K2}\times N1 \\ (K2,0) & \text{default} \end{cases}$$

$$l_\beta = max(l2_{temp}+1,2)$$

Step 3: Calculate the l_θ and l_Θ using the following case statement:

$$(l_\Theta, l_\theta) = \begin{cases} (l_\alpha, l_\alpha) & \text{if } (N2=0) \\ (l_\beta, l_\beta) & \text{if } (N1=0) \\ (l_\beta - (B\&b), l_\alpha) & \text{if } (l_\beta - l_\alpha = -1) \\ (l_\alpha - (B\&b), l_\beta) & \text{if } (l_\beta - l_\alpha = 1) \\ (l_\beta - 1, 0) & \text{if } (l_\beta - l_\alpha = 0) \\ (min(l_\alpha, l_\beta), min(l_\alpha, l_\beta)) & \text{default} \end{cases}$$

$$(l_\Theta, l_\theta) = \begin{cases} (l_\theta, l_\Theta) & \text{if } (S_{D2} * S_{N2} \neq S_{D1} * S_{N1}) \\ (l_\Theta, l_\theta) & \text{default} \end{cases}$$

Step 4: Calculate the $S_{\tilde{\theta}}$ and $S_{\tilde{\Theta}}$ using the following case statement:

$$S = \begin{cases} S_{D1} * S_{N1} & \text{if } (l_\beta - l_\alpha \geqslant 0 \;||\; N2 = 0) \\ S_{D2} * S_{N2} & \text{default} \end{cases}$$

$$(S_{\tilde{\Theta}}, S_{\tilde{\theta}}) = \begin{cases} (S, S) & \text{if } (N2=0) \\ (S, -S) & \text{if } (N1=0) \\ (S * Sign(l_\beta - l_\alpha), S) & \text{default} \end{cases}$$

Step 5: Calculate the $(c_{\tilde{\theta}}, s_{\tilde{\theta}})$ and $(c_{\tilde{\Theta}}, s_{\tilde{\Theta}})$ pairs using the following case statement:

$$\tilde{t_1} = \begin{cases} 0 & \text{if } l_\theta = 0 \\ 2^{-l_\theta} & \text{default} \end{cases}$$

$$\tilde{t_2} = \begin{cases} 0 & \text{if } l_\Theta = 0 \\ 2^{-l_\Theta} & \text{default} \end{cases}$$

$$(c_{\tilde{\theta}}, s_{\tilde{\theta}}) = \frac{1}{1+\tilde{t_1}^2} \times (1-\tilde{t_1}^2, 2\times S_{\tilde{\theta}} \times \tilde{t_1})$$

$$(c_{\tilde{\Theta}}, s_{\tilde{\Theta}}) = \frac{1}{1+\tilde{t_2}^2} \times (1-\tilde{t_2}^2, 2\times S_{\tilde{\Theta}} \times \tilde{t_2})$$

Step 6: Calculate the rotation matrices $\mathbf{R}_{\tilde{\theta}}$ and $\mathbf{R}_{\tilde{\Theta}}$ using the following case statement:

$$\mathbf{R}_{\tilde{\theta}} = \begin{cases} \begin{bmatrix} c_{\tilde{\theta}} & s_{\tilde{\theta}} \\ -s_{\tilde{\theta}} & c_{\tilde{\theta}} \end{bmatrix} & \text{if } S_{D1} < 0 \\ \begin{bmatrix} s_{\tilde{\theta}} & c_{\tilde{\theta}} \\ c_{\tilde{\theta}} & -s_{\tilde{\theta}} \end{bmatrix} & \text{else} \end{cases}$$

$$\mathbf{R}_{\tilde{\Theta}} = \begin{cases} \begin{bmatrix} c_{\tilde{\Theta}} & s_{\tilde{\Theta}} \\ -s_{\tilde{\Theta}} & c_{\tilde{\Theta}} \end{bmatrix} & \text{if } S_{D1} < 0 \\ \begin{bmatrix} s_{\tilde{\Theta}} & c_{\tilde{\Theta}} \\ c_{\tilde{\Theta}} & -s_{\tilde{\Theta}} \end{bmatrix} & \text{else} \end{cases}$$

Table 4.1: Different possibilities of choosing the fast rotation angles

$\tilde{\alpha}$ and $\tilde{\beta}$	Range	$\theta = \alpha - \beta$	$\tilde{\theta}_{big}$	$\tilde{\theta}_{small}$	$\tilde{\theta}$	$\Theta = \alpha + \beta$	$\tilde{\Theta}_{big}$	$\tilde{\Theta}_{small}$	$\tilde{\Theta}$
$\tilde{\alpha} = \frac{1}{2}$ $\tilde{\beta} = \frac{1}{2}$	Bb	$(\frac{-1}{4}, \frac{1}{4})$	$\frac{1}{4}$ or $\frac{-1}{4}$	0	0	$(1, \frac{6}{4})$	1	1	1
	Bs	$(0, \frac{3}{8})$	$\frac{1}{4}$	0	0	$(\frac{7}{8}, \frac{5}{4})$	1	1	1
	Sb	$(\frac{-3}{8}, 0)$	$\frac{-1}{4}$	0	0	$(\frac{7}{8}, \frac{5}{4})$	1	1	1
	Ss	$(\frac{-1}{8}, \frac{1}{8})$	$\frac{1}{8}$ or $\frac{-1}{8}$	0	0	$(\frac{3}{4}, 1)$	1	1	1
$\tilde{\alpha} = \frac{1}{2}$ $\tilde{\beta} = \frac{1}{4}$	Bb	$(\frac{1}{8}, \frac{1}{2})$	$\frac{1}{2}$	$\frac{1}{8}$	$\frac{1}{4}$	$(\frac{3}{4}, \frac{9}{8})$	1	1	1
	Bs	$(\frac{1}{4}, \frac{9}{16})$	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{4}$	$(\frac{11}{16}, 1)$	1	$\frac{1}{2}$	$\frac{1}{2}$
	Sb	$(0, \frac{1}{4})$	$\frac{1}{4}$	0	$\frac{1}{4}$	$(\frac{5}{8}, \frac{7}{8})$	1	$\frac{1}{2}$	$\frac{1}{2}$
	Ss	$(\frac{1}{8}, \frac{5}{16})$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{4}$	$(\frac{9}{16}, \frac{5}{4})$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
$\tilde{\alpha} = \frac{1}{2}$ $\tilde{\beta} = \frac{1}{8}$	Bb	$(\frac{5}{16}, \frac{5}{8})$	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{2}$	$(\frac{5}{8}, \frac{15}{16})$	1	$\frac{1}{2}$	$\frac{1}{2}$
	Bs	$(\frac{3}{8}, \frac{21}{32})$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$(\frac{19}{32}, \frac{7}{8})$	1	$\frac{1}{2}$	$\frac{1}{2}$
	Sb	$(\frac{3}{16}, \frac{3}{8})$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{2}$	$(\frac{1}{2}, \frac{11}{16})$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
	Ss	$(\frac{1}{4}, \frac{13}{32})$	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{2}$	$(\frac{17}{32}, \frac{5}{8})$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
$\tilde{\alpha} = \frac{1}{2}$ $\tilde{\beta} = \frac{1}{16}$	Bb	$(\frac{13}{32}, \frac{11}{16})$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$(\frac{9}{16}, \frac{27}{32})$	1	$\frac{1}{2}$	$\frac{1}{2}$
	Bs	$(\frac{7}{16}, \frac{45}{64})$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$(\frac{35}{64}, \frac{13}{16})$	1	$\frac{1}{2}$	$\frac{1}{2}$
	Sb	$(\frac{9}{32}, \frac{7}{16})$	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{2}$	$(\frac{7}{16}, \frac{19}{32})$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
	Ss	$(\frac{5}{16}, \frac{29}{64})$	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{2}$	$(\frac{27}{64}, \frac{9}{16})$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$

use B and b, and it will only affect one of the fast rotation angles in the TABLE 4.1. This case statement is represented in (4.18). We named this method ERFHSVD2. The $|| D ||$ of ERFHSVD2 vs. τ_1 and τ_2 is demonstrated in Figure 4.9. The comparison between the $|| D ||$ of ERFHSVD and ERFHSVD2 shows higher values for ERFHSVD which is expected since ERFHSVD2 is the less complex approximation. The RMS_{ODN} of both methods are represented in Figure 4.10. The loss in convergence speed is two extra rotation at maximum performance for 32 bit representation, while it requires an extra comparator. The performance of unquantized representation of ERFHSVD2 is also demonstrated in Figure 4.10 to prove that the floor in the RMS_{ODN} is due to the quantization and not approximations. This figure also demonstrates the difference between ERNSVD and ERFHSVD. The loss in convergence speed is four extra rotations at maximum performance for 32 bit representation, this difference is smaller when larger RMS_{ODN} is acceptable. We must note that depending on the implementation, one iteration of ERFHSVD might be equal

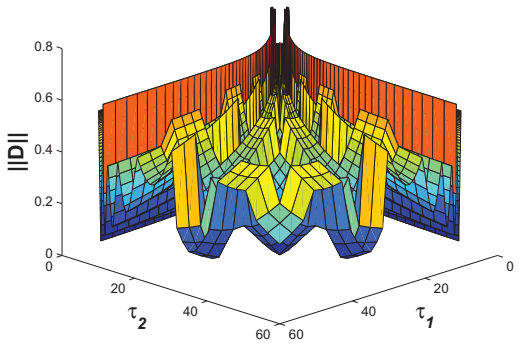


Figure 4.8: $||\ D\ ||$ vs. τ_1 and τ_2 for ERFHSVD

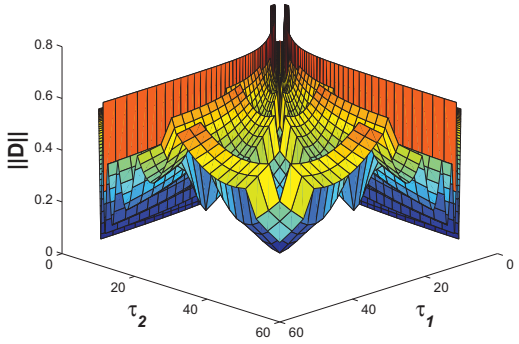


Figure 4.9: $||\ D\ ||$ vs. τ_1 and τ_2 for ERFHSVD2

to applying two iteration of ERNSVD. The ERNSVD needs to calculate and apply the symmetrizing rotation and then calculate and apply the diagonalizing rotation.

4.3 Hardware Implementation

These algorithms can be implemented in different methods on the higher level. Figure 5.9 shows a high level systolic implementation of the decomposition algorithm for an 8×8 matrix. Figure 4.12 shows how the scheduling for this implementation can be or-

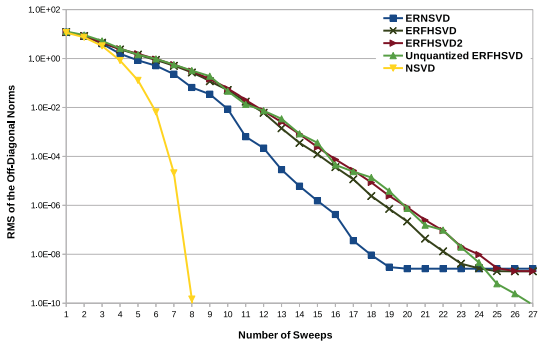


Figure 4.10: Comparison of the RMS_{ODN} for NSVD, ERNSVD, ERFHSVD, Unquantized ERFHSVD, and ERFHSVD2

ganized so four independent rotations are applied in each clock cycle. Each pair shows the number of rows and columns that rotation is calculated from and applied to. While different high level designs choose different method to manage their memory unit, timing, and connections, majority of these architectures follow some common design footsteps. These architectures are constructed from two different types of processor: diagonal and non-diagonal processors. The diagonal processor calculate, transmit and applies the horizontal and vertical rotations while the non-diagonal processor receives, and applies the rotations. Since all our contributions can be explained with more details in lower design discussions, we focus on the design of basic circuits for diagonal and non-diagonal processors (DP, and NDP). The discussed and presented are only one possible implementation. The two's complement representation is used in this deign to represent negative numbers. The fully combinational implementation of the design is discussed here.

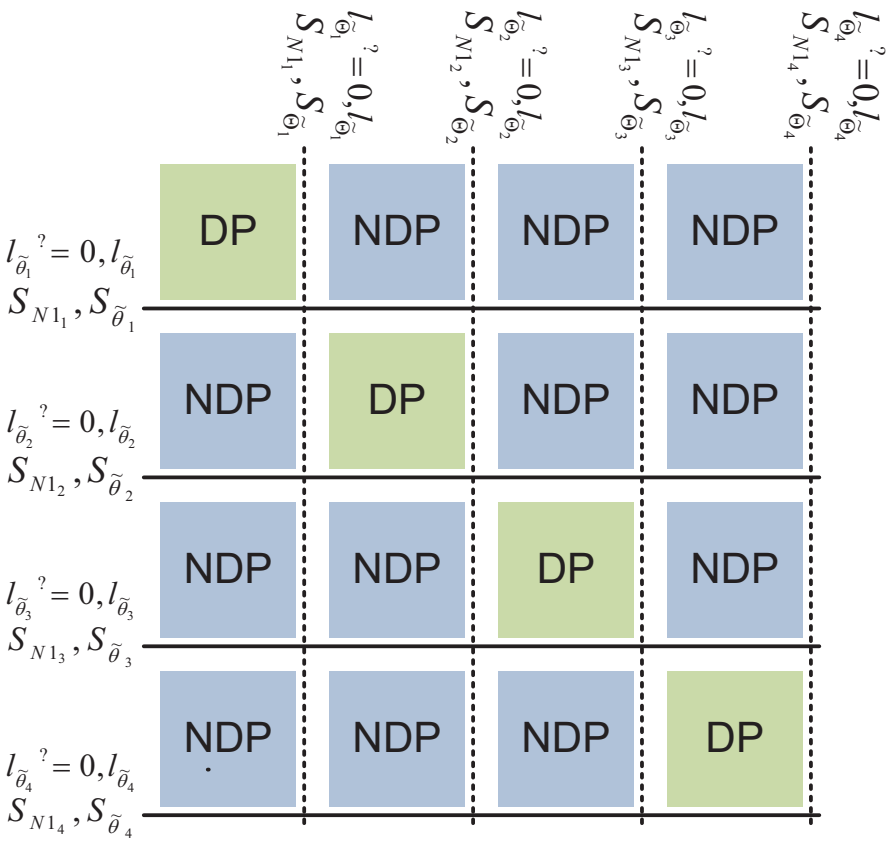


Figure 4.11: Architecture of the design.

$$
\begin{aligned}
(p,q) = &(1,2)(3,4)(5,6)(7,8) \\
&(1,4)(2,6)(3,8)(5,7) \\
&(1,6)(4,8)(2,7)(3,5) \\
&(1,8)(6,7)(4,5)(2,3) \\
&(1,7)(8,5)(6,3)(4,2) \\
&(1,5)(7,3)(8,2)(6,4) \\
&(1,3)(5,2)(7,4)(8,6)
\end{aligned}
$$

Figure 4.12: Scheduling of matrix processing order.

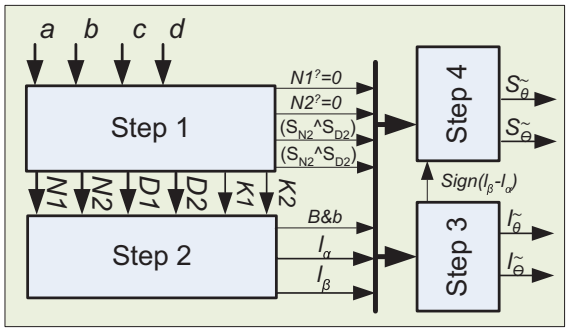


Figure 4.13: Diagram for calculating the Given's Rotations based on the ERFHSVD algorithm

4.3.1 Calculating the Rotations

Figure 4.13 demonstrates the diagram for calculating the Given's Rotations based on the ERFHSVD algorithm. This involves the first four steps of Algorithm 7. The inputs are the elements of 2×2 matrix that is the target of decomposition. The outputs are the sign ($Sign$) and power (l) in $tan(\tilde{x}) = \tilde{t} = Sign \times 2^{-l}$ for both rotation angles $\tilde{\theta}$ and $\tilde{\Theta}$. This circuit is only exists in diagonal processors (DPs). After calculating the value this circuit transmits the values to the circuit for applying rotations in the DP. This circuit also sends the required signals to the circuit for applying rotations in the same DP ($l_{\tilde{\Theta}}$, $l_{\tilde{\theta}}$, $S_{\tilde{\Theta}}$, $S_{\tilde{\theta}}$,S_{N1}, $l^?_{\tilde{\Theta}} = 0$, and $l^?_{\tilde{\theta}} = 0$) and NDPs in the same row ($l_{\tilde{\Theta}}$, $S_{\tilde{\Theta}}$, S_{N1}, and $l^?_{\tilde{\Theta}} = 0$)[1], and column ($l_{\tilde{\theta}}$, $S_{\tilde{\theta}}$, S_{N1}, and $l^?_{\tilde{\theta}} = 0$). The circuit for each of the steps is discussed hereafter.

[1] $l^?_{\tilde{\Theta}} = 0$ is the signal name and will be one if $l_{\tilde{\Theta}}$ is equal to zero (this signal is used to make decision on cases in **Step 2** of the **Algorithm 7**), and S_{N1} is the sign of the numerator in equation (5.11).

ERFHSVD Step 1

Figure 4.14 demonstrates the diagram of the proposed design for the first step of ERFHSVD algorithm. This circuit generates the initial values for being used at the next steps in ERFHSVD algorithm. The inputs are the elements of 2×2 matrix that is the target of decomposition. The MSB block does not have any cost in VLSI implementation and it is demonstrating that the most significant bit of the value is used to determine the sign (The implementation is two's complement). The blocks with $|\blacklozenge|$ on them are the circuits to calculate the absolute value of the input value. We assume this will cost an XOR complimenting circuit and an adder to calculate the two's complement of negative numbers. To avoid overflow or the need to use saturation, the adder size of the $|\blacklozenge|$ circuits should be at least of the same size as the matrix input argument. The P-Enc blocks are priority encoders as explained in (4.12), where v is the valid signal and is zero when the input signal is equal to zero, and it is one for rest of the cases. The blocks with $<<\ 1$ are indicating shifts to the left (or multiplying by two) and their VLSI implementation does not have any cost. The last two subtractors in this diagram have the size of $ceil(log_2(bit))$ where the bit is the number of bits each element of input matrix is represented with. One of the conditions of the second step is to limit the value of $K1$ and $K2$ to the minimum of two. The over flow outputs of the last two subtractors are indicating a negative result since both inputs are positives.

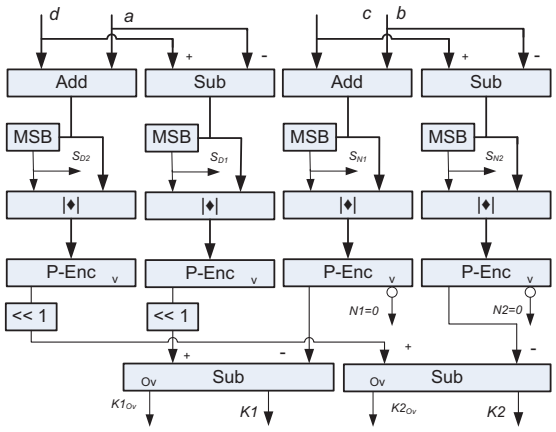


Figure 4.14: Diagram of the first step of ERFHSVD algorithm

ERFHSVD Step 2

Figure 4.15 demonstrates the diagram of the proposed design for the second step of ERFHSVD algorithm. This block shows how l_α is calculated. A similar block can be used to generate l_β. This circuit uses an adder and a B-Shifter to generate the signals required for the case statement in **Step 2**. The blocks with $<<\ 1$ are indicating shifts to the left (or multiplying by two) and their VLSI implementation does not have any cost. The adder, B-Shifter, and comparators are of the size *bit*. The last adder and mux in the block diagram are of the size $ceil(log_2(bit))$. The L-Ckt is indicating a logical circuit that can be implemented with a and, or, and inverter representation or any other means necessarily. We have merged the case statements of **Step 2** with the mathematics phrase comeing right after them and represented them both in one circuit. In fact the last multiplexer in the diagram is saturating the results to the minimum of two; while the "L-Ckt 2" is generating its select signal inputs. The "L-Ckt 1" outputs, zero, one or two to be added in the final adder based on the case

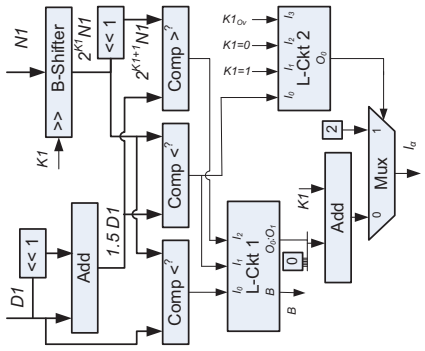


Figure 4.15: Diagram of the second step of ERFHSVD algorithm

condition; while also generating the "B" signal. For "L-Ckt 1", $B = I_0\overline{I_1} + I_2$, $O_0 = \overline{I_1}\ \overline{I_2}$, and $O_1 = I_2$. For "L-Ckt 2", $O_0 = I_3 + I_2 + I_1I_0$. The signal $K1 = 0$ is generated with an eight-input NOR gate while the signal $K = 1$ is generated with a NOR gate and an inverter. We did not show these gates in the figure in favor of keeping the diagrams straightforward.

ERFHSVD Step 3

Figure 4.16 demonstrates the diagram of the proposed design for the third step of ERFHSVD algorithm. This circuit takes the l_α, l_β, $N2^? = 0$, Bb, $(S_{N2} \oplus SD2) \oplus (S_{N1} \oplus S_{D1})$, $N1^? = 0$[2] signals as inputs and outputs the values of $l_{\tilde{\theta}}$ and $l_{\tilde{\Theta}}$ to the circuit for applying rotations and the sign bit of $l_\beta - l_\alpha$ to **Step 4**. This design merges the case statement in **Step 3** and the mathematical phrase after it, and implements both together. Shift left logical (SLL blocks marked with $<<$), adder, subtractor, and multiplexers blocks are of the size $ceil(log_2(bit))$. The block that is supposed to determine if the result of the subtract is zero (this block is marked with $^? = 0$) requires a NOR gate of size $ceil(log_2(bit)) + 1$ assuming that overflow (Ov) can happen and for detecting "one" an inverter and a NOR gate is re-

[2] $N1^? = 0$ will be one if $N1 = 0$ (which $N1$ is the numerator in equation(5.11)).

quired, since both l_α and l_β are positive an AND gate of size $ceil(log_2(bit))$ with an inverter can be used to synthesize the block marked with $^? = -1$. This provision is taken to prevent the usage of comparators. The L-Ckt 1 applies any change necessary on the value of l_β by adding zero, minus one, or minus two to it. The L-Ckt 2 generates the input signals to the mux based on the inputs to assure that correct values are assigned to $l_{\tilde{\theta}}$, and $l_{\tilde{\Theta}}$. Since $l_\beta \geq 2$, the result of the additions can not be negative, the Ov outputs of the adders can be ignored. The final multiplexer of the circuit is needed to guarantee that diagonalized output matrix is normalized. In L-Ckt 1, $O_0 = \overline{I_0}\ \overline{I_1}\ \overline{I_2}\ \overline{I_5} + \overline{I_0}\ I_2\ \overline{I_4}\ \overline{I_5} + \overline{I_0}\ I_1\ I_4\ \overline{I_5}$, $O_1 = \overline{I_1}\ I_4\ \overline{I_5} + \overline{I_0}\ \overline{I_1}\ \overline{I_5}$, $O_2 = \overline{I_0}\ \overline{I_1}\ \overline{I_2}\ I_5 + \overline{I_0}\ I_2\ \overline{I_4}\ I_5 + \overline{I_0}\ I_1\ I_4\ I_5$, and $O_3 = \overline{I_1}\ I_4\ I_5 + \overline{I_0}\ \overline{I_1}\ I_5$. In L-Ckt 2, the signals are defined as follows: $O_0 = I_5\ I_3 + I_5\ \overline{I_0}\ I_1 + I_6$, $O_1 = I_0 + I_1 + I_2 + I_3\overline{I_5} + I_6$, $O_2 = I_3\overline{I_5} + \overline{I_0}\ I_1\ \overline{I_5} + I_6$, and $O_3 = I_0 + I_1 + I_2 + I_5\ I_3 + I_6$.

Using the circuit presented in Figure 4.16 with minor changes in L-Ckt 1 and L-Ckt 2 implementation, every different value in the first three rows of TABLE 4.1 can be assigned as the rotation angle. If implementation of more rows from the table is required, the circuit to detect plus and minus two also should be added and fed to the logical circuits. The simple design presented for this step of the algorithm would assist the designers to change table values based on their application need. For more complex design, in case the number of inputs to the logical circuits is high, an alternative design can be explored that two-input multiplexers (two AND gates of size $ceil(log_2(bit))$) are used to impediment the second case statement in **Step 3**. The multiplexer swaps the value of $l_{\tilde{\theta}}$, and $l_{\tilde{\Theta}}$ if $(S_{D1} \oplus S_{N1}) \oplus (S_{D2} \oplus S_{N2}) = 1$. This will reduce the complexity of L-Ckt 1 and might eliminate the need

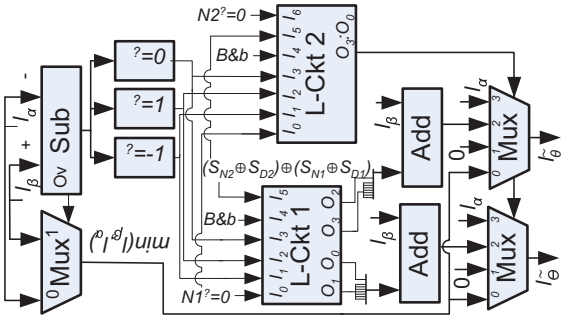


Figure 4.16: Diagram of the third step of ERFHSVD algorithm

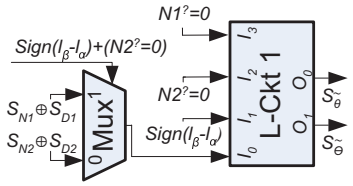


Figure 4.17: Diagram of the fourth step of ERFHSVD algorithm

for one of the adders.

ERFHSVD Step 4

Figure 4.17 demonstrates the diagram of the proposed design for the fourth step of ERFHSVD algorithm. This circuit takes the sign bit of $l_\beta - l_\alpha$ ($Sign(l_\beta - l_\alpha)$), $N2^? = 0$, $N1^? = 0$, S_{N1}, S_{D1}, S_{N2}, and S_{D2} as input and generates the sign of the rotations ($S_{tilde\Theta}$ and $S_{tilde\theta}$). Correct implementation of this step is vital for convergence of the iterations as well as achieving the normalized diagonal elements. For "L-Ckt 1", $O_0 = \overline{I_0}\,\overline{I_2} I_3 + I_0 \overline{I_2}\,\overline{I_3} + I_0 I_2$, and $O_1 = \overline{I_0} I_1 \overline{I_2}\,\overline{I3} + I_0 I_3 + I_0 I_2 + I_0 \overline{I_1}$.

4.3.2 Applying the Rotations

Figure 5.8 demonstrates the diagram of the proposed design for applying rotations. This circuit includes the major part of the NDP (except memory units and the mechanism to re-

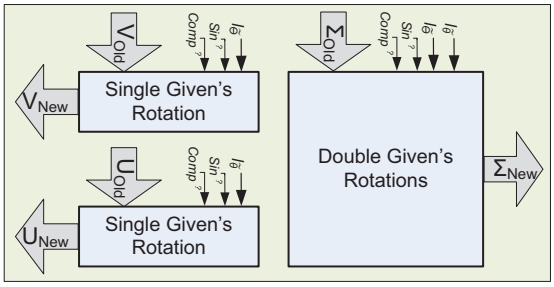


Figure 4.18: Diagram of the circuit for applying rotations

ceive the input arguments and transfer the results). Assuming a 2×2 matrix decomposition is presented in (4.19). For an iterative algorithm this unit should be able to apply both rotation matrices on the old value of $\mathbf{\Sigma}$ ($\mathbf{\Sigma}_{Old}$) based on the values of $l_{\tilde{\theta}}$ and $l_{\tilde{\Theta}}$, it also should apply one rotation matrix on the old value of $\mathbf{U}$ ($\mathbf{U}_{Old}$) based on the values of $l_{\tilde{\theta}}$ and one rotation matrix on the old value of $\mathbf{V}$ ($\mathbf{V}_{Old}$) based on the values of $l_{\tilde{\Theta}}$ to generate the new values. The $Sin^?$[3] signal is in fact one if $S_N \geq 0$ as we will explain in the next sub section. The initial value of $\mathbf{\Sigma}$ is $\mathbf{A}$ and the initial value of $\mathbf{U}$ and $\mathbf{V}$ is identity matrix.

$$\mathbf{A} = \mathbf{U} \times \mathbf{\Sigma} \times \mathbf{V}^T \tag{4.19}$$

Applying Double and Single Given's Matrix Rotations

Figure 4.19 demonstrates the diagram of the proposed design for calculating the single Given's Rotation. It is important to note that applying a Given's rotation is in fact a multiplication of two 2×2 matrices. This is equal to eight multiplication and four addition if no other consideration is made about the implementation of the system. We can use this

[3] $Sin^?$ is one when a block going to apply sine rotation and cosine rotation otherwise.

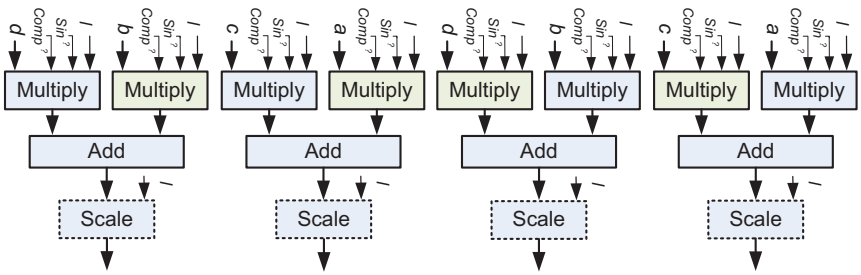


Figure 4.19: Diagram for applying single Given's rotation

circuit to calculate the value of V and U. The scale circuit is demonstrated with doted outline to remind readers that its presence or implementation complexity (and accuracy as the result) can be adapted based on the application need. The $Sin^{?}$ signal that is an input to each Multiply block shows if that block has to multiply the input argument with $Sin(\tilde{x})$ or $Cos(\tilde{x})$. A control unit can separately assign different values to each Multiply block, or it can choose to multiply the input value with $Sin(\tilde{x})$ or $Cos(\tilde{x})$. In the design the blocks are with a different color to make sure they multiply the input argument with $Cos(\tilde{x})$ while the normal blocks are multiplying the input arguments with $Sin(\tilde{x})$.

Figure 4.20 demonstrates the diagram of the proposed design for calculating the single Given's Rotations. The same notation as in Fig 4.19 is used in this diagram. The circuit in this diagram has double complexity compared to the design of Figure 4.19. This is expected since this circuit has to multiply the input 2×2 matrix with two rotation matrices of $R_{\tilde{\theta}}$ and $R_{\tilde{\Theta}}$.

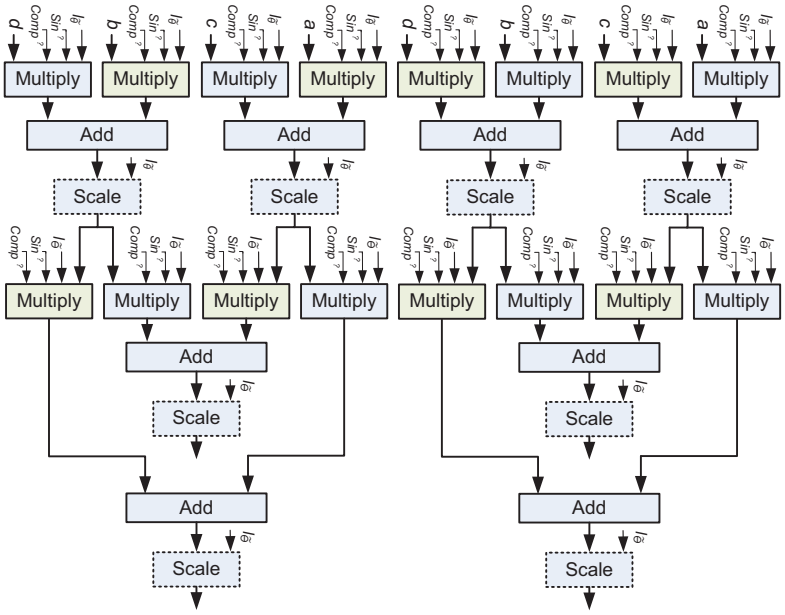


Figure 4.20: Diagram for applying double Given's rotations

Multiplying and Scaling

The rotation angles are the approximations to $t = |tan(x)|$ with $\tilde{t} = |tan(\tilde{x})| = 2^{-l}$ and as the result of applying double rotations the elements of any rotation matrix should be zero, one, or the values derived from equation (4.20) or equation (4.21). The *Sign* in (4.21) will be determined in **Step 4** ($S_{\tilde{\theta}}$, *or* $S_{\tilde{\Theta}}$). The part $\frac{1}{1+\tilde{t}^2}$ is common between all the coefficients and we will discuss it in detail after presenting the circuit which applies the uncommon part of the multiplications (this is refereed to as scaling) if the proposed circuit is able to apply any of these coefficients using only control signals then we can merge **Step 5** and **Step 6** in algorithm 7 (this is refereed to as multiplying).

$$\frac{1}{1+\tilde{t}^2} \times 1 - \tilde{t}^2 \tag{4.20}$$

$$\frac{1}{1+\tilde{t}^2} 2 \times Sign \times \tilde{t} \tag{4.21}$$

Figure 4.21 demonstrates the diagram of the proposed design for multiplications. This is a fully combinational circuit and a possible design. The input A is any of the elements of the matrix in (5.10). If ξ is zero ($l = 0,\ Sin^? = 1,\ Comp^? = 1$), then the adder in the figure is adding A with $-A$. If ξ is one ($l = 0,\ Sin^? = 0,\ Comp^? = 0$), then the adder is adding A with zero. To apply equation (4.20), the input value has to be shifted to the right twice the value of l and then subtracted from the original value ($l \neq 0,\ Sin^? = 0,\ Comp^? = 1$). The circuit should be able to apply equation (4.21) assuming *sign* can be negative or positive. For this equation the adder only adds one to the input value to convert one's compliment values to two's compliment values ($l \neq 0,\ Sin^? = 1,\ Comp^? = 1$). For this circuit a control unit can define the values for selecting input of multiplexers or the value of $Comp^?$[4]; however if we use the signals as defined in Figure 4.21, we can assign $Sin^? = S_{N1}$ and just swap the input index for multiplexers if a unit needs to apply $Cos(\tilde{x})$ when $S_{N1} < 0$. If the multiplication circuit belongs to a DP then the S_{N1} signal of the same processor is used to determine the $Sin^?$ signals, while if the circuit belongs to an NDP then S_{N1} that is transmitted from the DP in the same row is used for determining the $Sin^?$ signals in the first row of the double Given's rotations circuit and updating the values of $\mathbf{U}$. The S_{N1} that is transmitted from the DP in the same column is used for determining the $Sin^?$ signals in the second row of the

[4] $Comp^?$ is one if the multiplicand is negative.

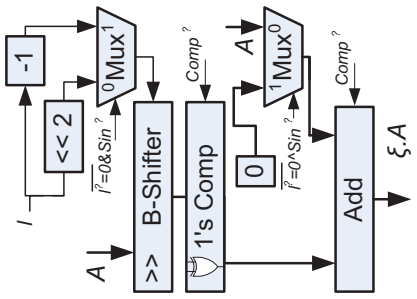


Figure 4.21: Diagram of the multiplier for ERFHSVD algorithm

double Given's rotations circuit and updating the values of **V**.

Applying scaling is one of the more resource demanding parts in this design while it might not be necessary for every application to apply scaling coefficients when decomposing a matrix. This generally results in applying orthogonal rotation angles and an increase in the value of diagional elements of decomposition which might be acceptable for some applications. Comparing the fast rotations method with CORDIC method the complexity of the scaling circuit is increased when applying fast rotations. The fact that CORDIC method applies all the rotation angles and only the sign of the rotations are different results in a constant scaling value; while, the fast rotations can be different any time, this results in different scaling for different rotation angels. [6] and [7] suggest using the Tailor series representation of $\frac{1}{1+\tilde{t}^2}$ as in (4.22). The estimation of complexity depends on the implementation, but a simplified look at the problem is presented in [6] and [7]. The complexity of other CORDIC based implementations is also presented there. The implementation of the scaling factor for 32 bits implementation requires four Shifts and four additions. A closer look at the scaling coefficient would help reducing the complexity of scaling circuit to four addition, and one shift. Table 4.2 shows the scaling coefficient that needs to

Table 4.2: Scaling values for different rotation angles

Scale (λ)					Circuit representation of $\lambda.G$ (any 32 bit input)
l	-2l	Fraction	Decimal	Binary (32 bit)	
1	2	$\frac{1}{1+\frac{1}{4}}$	$\frac{4}{5}$	0.11001100110011001100110011001100	$Acc.\{Acc.\{Acc.\{G-2^{-2}G\}>>4\}>>8\}>>16$
2	4	$\frac{1}{1+\frac{1}{16}}$	$\frac{16}{17}$	0.11110000111100001111000011110000	$Acc.\{Acc.\{G-2^{-4}G\}>>8\}>>16$
3	6	$\frac{1}{1+\frac{1}{64}}$	$\frac{64}{65}$	0.11111100000011111100000011111100	$Acc.\{G-2^{-6}G\}>>12+(G-2^{-6}G)>>24$
4	8	$\frac{1}{1+\frac{1}{256}}$	$\frac{256}{257}$	0.11111111000000001111111100000000	$Acc.\{G-2^{-8}G\}>>16$
5	10	$\frac{1}{1+\frac{1}{1024}}$	$\frac{1024}{1025}$	0.11111111110000000000111111111100	$Acc.\{G-2^{-10}G\}>>20$
6	12	$\frac{1}{1+\frac{1}{4096}}$	$\frac{4096}{4097}$	0.11111111111100000000000011111111	$Acc.\{G-2^{-12}G\}>>24$
7	14	$\frac{1}{1+\frac{1}{16384}}$	$\frac{16384}{16385}$	0.11111111111111000000000000001111	$Acc.\{G-2^{-14}G\}>>28$
8	16	$\frac{1}{1+\frac{1}{65536}}$	$\frac{65536}{65537}$	0.11111111111111110000000000000000	$G-2^{-16}G$
9	18	$\frac{1}{1+\frac{1}{262144}}$	$\frac{262144}{262145}$	0.11111111111111111100000000000000	$G-2^{-18}G$
10	20	$\frac{1}{1+\frac{1}{1048576}}$	$\frac{1048576}{1048577}$	0.11111111111111111111000000000000	$G-2^{-20}G$
11	22	$\frac{1}{1+\frac{1}{4194304}}$	$\frac{4194304}{4194305}$	0.11111111111111111111110000000000	$G-2^{-22}G$
12	24	$\frac{1}{1+\frac{1}{16777216}}$	$\frac{16777216}{16777217}$	0.11111111111111111111111100000000	$G-2^{-24}G$
13	26	$\frac{1}{1+\frac{1}{67108864}}$	$\frac{67108864}{67108865}$	0.11111111111111111111111111000000	$G-2^{-26}G$
14	28	$\frac{1}{1+\frac{1}{268435456}}$	$\frac{268435456}{268435457}$	0.11111111111111111111111111110000	$G-2^{-28}G$
15	30	$\frac{1}{1+\frac{1}{1073741824}}$	$\frac{1073741824}{1073741825}$	0.11111111111111111111111111111100	$G-2^{-30}G$
16	32	$\frac{1}{1+\frac{1}{4294967296}}$	$\frac{4294967296}{4294967297}$	0.11111111111111111111111111111111	$G-2^{-32}G\approx G$

be applied for each rotation angle of $tan(\tilde{x}) = 2^{-l}$. Increase in the value of l causes the scaling coefficient to merge to one; as the result, for $l \geq 16$ the 32 bit representation of the coefficient is rounded to one. The notation $Z = Acc.\{\Delta\} \gg \delta$ is equal to $Z = \Delta + \Delta >> \delta$ which $\Delta \gg \delta$ means arithmetic shift of the Δ to right δ times. Figure 4.22 demonstrates the diagram of the proposed design for scaling in ERFHSVD algorithm.

$$\frac{1}{1+\tilde{t}^2} = (1-2^{-2l})(1+2^{-4l})(1+2^{-8l})(1+2^{-16l})... \tag{4.22}$$

In Figure 4.22 a 32 bit value is multiplied by scale factor λ based on the representation in the last column of table 4.2. This circuit tries to gain the benefit from repetitive nature of

coefficient λ. If $l \geq 16$ then the scaling factor would simply be one. The circuit presented in this figure is not extendable directly for higher number of bits accuracy and it would be a simpler circuit for lower number of bits. The select signals for the multiplexers has to be assigned based on the value of l. For the multiplexer we assumed that the three and four-input multiplexers are made of two-input multiplexers and rearrange them in a way that minimum number of two-input multiplexers are required. In addition, we assume that the Barrel shifters are made out of two-input multiplexers to be able to compare the complexity of the new implementation with the implementation that requires four shifts and four adds. Following the assumptions made, the shifts are requiring the use of B-Shifter since the value of l is different each time. Each B-Shifter will require 160 two-input multiplexers, and our design will require 224 two-input multiplexers for the multiplexers represented in the circuit. This will result in saving 156 two-input multiplexers. This results is approximately saving the cost of one B-Shifter. This circuit also has effects on critical path delay. While a B-Shifter that uses 160 two-input multiplexer has five level of two-input multiplexers the multiplexers in our design at most have two levels. The major benefit of the circuit presented in Figure 4.22 is when some error occurs in applying the scaling factor is acceptable. If we implement the circuit only with the first subtraction, the maximum error in applying the scale coefficient will be 6.25%. If we represent the circuit with the second adder the maximum error in scaling coefficient will be 0.39% (it multiplies the input value with 0.797 instead of 0.8). The circuit with three adders would have maximum of 0.024% error in applying the coefficients (it multiplies the input value with 0.7998 instead of 0.8).

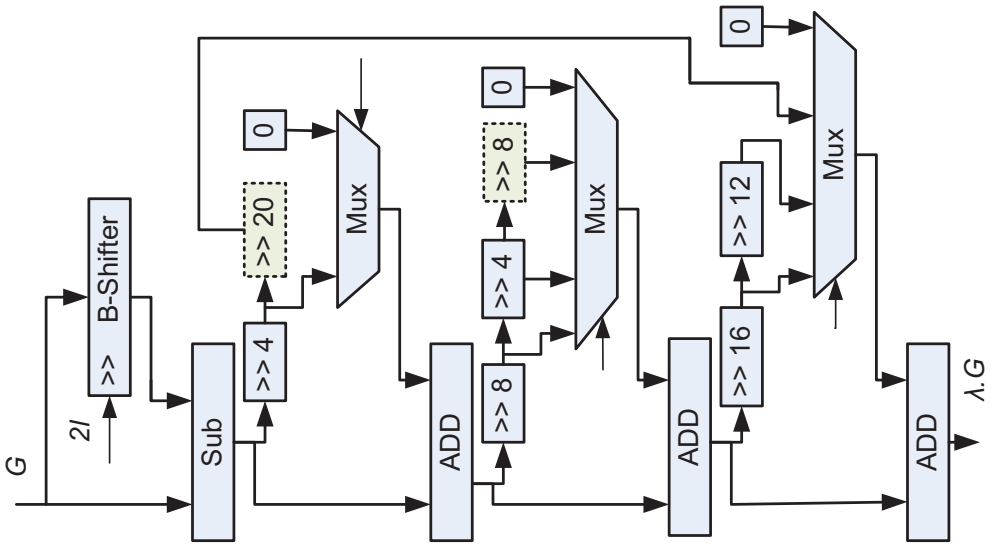


Figure 4.22: Diagram of the scaling circuit for ERFHSVD algorithm

Due to the complexity of scaling circuit in comparison with the rest of the architecture and its variable importance due to the application, we recommend a detailed study for each application. Based on the required accuracy of the result the scaling circuit may be completely ignored or represented with lees complex and less accurate circuit. For example the two shifts that are represented in different color (and doted exterior line), their associated paths, and the last level of adder/multiplexer are only need to be implemented if 32 bit accuracy for the scaling coefficient is needed.

4.4 Complexity

The calculation of computational complexity (delay, and resource requirement) depends on the multiple factors other than only the block diagram of the subsystem; however we will try to provide an approximation using some basic assumption to provide a better un-

derstanding of this work achievements. We divide the discussions to two sub sections of calculating and applying the rotations. We assume the critical path of any of the blocks in the block diagram representation of any circuit is known as it is represented with symbol Δ. We assume that every subtractor (Sub) is an adder with carry-in equal to one and an array of inverters; as the result, its delay is higher than an adder of same number of bits. We assume that the size of the inputs are Λ bits with $\lambda = ceil(log_2\Lambda)$.

For resource estimation, we use A as area or resources required for implementing of a particular circuit. The $A_{\Lambda bitAdd}$ is representing the area or resources required to implement an adder of size Λ bits.

4.4.1 Calculating the Rotations

In order to be able to evaluate the delay of the system we assume that an adder of size Λ bits has higher delay than a B-Shifter of the same size.

Step 1

The critical path starts from the subtractor that generates the absolute value of $D1$ or $D2$ in the first step and the adder that generates $1.5 \times D1$ in the second stage, instead of the adder that generates the $N1$ or $N2$ in the first Stage and the B-Shifter in the second stage. In (4.23) the critical path delay for the first step of ERFHSVD algorithm is presented.

$$\begin{aligned}\Delta_{Step\ 1} = \Delta_{\Lambda bitSub} + \Delta_{\Lambda bit|\blacklozenge|} + \Delta_{\Lambda bitP-Enc} \\ + \Delta_{\lambda bitSub}\end{aligned} \tag{4.23}$$

The phrase $\Delta_{\Lambda bitSub}$ means that the critical path of a Λ bit subtractor; similar notations are used for other elements. The $|\blacklozenge|$ is the circuit calculating the absolute values, and $\Delta_{\Lambda bitP-Enc}$ refers to the critical path of a Λ bit P-Enc. As explained in the beginning of this section the delay path of the subtractor and $|\blacklozenge|$ can be translated to the delay of adders. In (4.24) we tried to apply this.

$$\begin{aligned}\Delta_{Step\ 1} &= \Delta_{\Lambda bitAdd} + \Delta_{Neg} + \Delta_{\Lambda bitAdd} \\ &+\Delta_{Compelement} + \Delta_{\Lambda bitP-Enc} + \Delta_{\lambda bitAdd} + \Delta_{Neg} \\ &= 2\times\Delta_{\Lambda bitAdd} + 2\times\Delta_{Inv} + \Delta_{2InpXOR} \\ &+\Delta_{\Lambda bitP-Enc} + \Delta_{\lambda bitAdd}\end{aligned} \tag{4.24}$$

We assume that Δ_{Neg} refers to the delay of an inverter and $\Delta_{Compelement}$ refers to the delay of a two-input XOR gate. We replaced the Δ_{Neg} with Δ_{Inv} (inverter) and $\Delta_{Compelement}$ with $\Delta_{2InpXOR}$ (two-input XOR) and we will keep that notation hereafter. The Area required to implement this circuit is represented in (4.25).

$$\begin{aligned}A_{Step\ 1} &= 8\times A_{\Lambda bitAdd} + 2\times(\Lambda+\lambda)\times A_{Inv} \\ &+4\times\Lambda\times A_{2InpXOR} + 4\times\Delta_{\Lambda bitP-Enc} \\ &+2\times A_{\lambda bitAdd}\end{aligned} \tag{4.25}$$

Step 2

In (4.27) the Critical path delay for the first step of ERFHSVD algorithm is presented. In this equation $\Delta_{2InpMux}$ is the critical path delay of a two-input multiplexer.

$$\begin{aligned}\Delta_{Step\ 2} &= \Delta_{\Lambda bitADD} + \Delta_{Complement} + \Delta_{L-Ckt\ 1} \\ &\quad + \Delta_{\lambda bitAdd} + \Delta_{2InpMux} \\ &= \Delta_{\Lambda bitADD} + \Delta_{2InpXOR} + \Delta_{L-Ckt\ 1} \\ &\quad + \Delta_{\lambda bitAdd} + \Delta_{2InpMux}\end{aligned} \tag{4.26}$$

Where,

$$\Delta_{L-Ckt\ 1} = \Delta_{Inv} + \Delta_{2InpAnd} + \Delta_{2InpOr} \tag{4.27}$$

The Area required to implement this circuit is represented in (4.30).

$$\begin{aligned}A_{Step\ 2} &= 2 \times (A_{\Lambda bitAdd} + A_{\Lambda bitShifter} \\ &+ 3 \times A_{\Lambda bitComparetor} + A_{L-Ckt\ 1} + 4A_{L-Ckt\ 2} + \\ &\quad A_{\lambda bitAdd} + \lambda \times A_{2InpMux})\end{aligned} \tag{4.28}$$

Where,

$$A_{L-Ckt\ 1} = 3 \times (A_{Inv} + A_{2InpOr} + 2 \times A_{2InpAnd} \tag{4.29}$$

and

$$A_{L-Ckt\ 2} = A_{3InpOr} + A_{2InpAnd} \tag{4.30}$$

Step 3

In (4.33) the Critical path delay for the first step of ERFHSVD algorithm is presented. In this equation $\Delta_{2InpMux}$ is the critical path delay of a two-input multiplexer.

$$\begin{aligned}\Delta_{Step\ 3} &= \Delta_{\lambda bitSub} + \Delta_{\lambda+1 InpNor} + \Delta_{L-Ckt\ 1} \\ &\quad + \Delta_{\lambda bitAdd} + \Delta_{4InpMux} \\ &= 2 \times \Delta_{\lambda bitAdd} + \Delta_{\lambda+1 InpNor} + \Delta_{L-Ckt\ 1} \\ &\quad + \Delta_{Inv} + \Delta_{4InpMux}\end{aligned} \tag{4.31}$$

If we assume the four-input multiplexer is made of two levels of two-input multiplexers, we can further simplify the equation.

$$\begin{aligned}\Delta_{Step\ 3} &= 2 \times \Delta_{\lambda bitAdd} + \Delta_{\lambda+1 InpNor} + \Delta_{L-Ckt\ 1} \\ &\quad + \Delta_{Inv} + 2 \times \Delta_{2InpMux}\end{aligned} \tag{4.32}$$

where,

$$\Delta_{Ckt\ 1} = \Delta_{Inv} + \Delta_{3InpOr} + \Delta_{4InpAnd} \tag{4.33}$$

The Area required to implement this circuit is represented in (4.34). We assume that nine-input NOR and AND gates do require the same area.

$$\begin{aligned}A_{Step\ 3} &= 3 \times A_{\lambda bitAdd} + 3\ timesA_{(\lambda+1)InpNor} \\ &\quad + A_{L-Ckt\ 1} + A_{L-Ckt\ 2} + 10 \times A_{Inv} \\ &\quad + \lambda \times A_{4InpMux} + \lambda \times A_{2InpMux}\end{aligned} \tag{4.34}$$

If we assume that a four-input multiplexer, is made of three, two-input multiplexers the area requirement is presented in (4.39).

$$\begin{aligned} A_{Step\ 3} = 3 \times A_{\lambda bitAdd} + 3 \times A_{\lambda+1 InpNor} \\ + A_{L-Ckt\ 1} + A_{L-Ckt\ 2} + 10 \times A_{Inv} \\ + \Lambda \times A_{2InpMux} \end{aligned} \tag{4.35}$$

Where,

$$\begin{aligned} A_{Ckt\ 1} = 6 \times A_{Inv} + 2 \times A_{3InpOr} + 2 \times A_{2InpOr} \\ + 4 \times A_{3InpAnd} + 6 \times A_{4InpAnd} \end{aligned} \tag{4.36}$$

and,

$$\begin{aligned} A_{Ckt\ 2} = 2 \times A_{Inv} + 2 \times A_{3InpOr} + 2 \times A_{5InpOr} \\ + 3 \times A_{2InpAnd} + 2 \times A_{3InpAnd} \end{aligned} \tag{4.37}$$

Step 4

In (4.38) the area requirement of the proposed circuit for the fourth step of ERFHSVD is estimated. The Critical path delay of this circuit is not presented since the delay of that circuit does not have any affect on the total delay of the design. In fact the delay of this step is less than the delay of the third step of the algorithm which runs in parts parallel to this step of the ERFHSVD algorithm.

$$A_{Step\ 4} = A_{2InpMux} + A_{L-Ckt\ 1} + A_{2InpXOR} \tag{4.38}$$

where,

$$A_{Ckt\ 1} = timesA_{4InpOr} + imesA_{3InpOr} + 3 \times A_{3InpAnd} + 4 \times A_{2InpAnd} \tag{4.39}$$

4.4.2 Applying the Rotations

The circuit of applying the rotations has expanding symmetry and is self-similar. This simplifies its implementation and complexity estimation. This also makes it a good candidate for pipelining. A high level look at the design of this circuit shows that it is made of 4 similar blocks each capable of applying single Given's Rotation. The critical path is determined by the circuit that applies double Given's rotations.

$$\Delta_{ApplyingRotations} = 2 \times \Delta_{OneGiven'sRotation} = 2 \times (\Delta_{\Lambda bitMultiply} + \Delta_{\Lambda bitAdd} + \Delta_{\Lambda bitScale}) \tag{4.40}$$

$$A_{ApplyingRotations} = 4 \times A_{OneGiven'sRotation} = 4 \times (\lambda \times A_{\Lambda bitMultiply} + 4 \times A_{\Lambda bitAdd} + 4 \times A_{\Lambda bitScale}) \tag{4.41}$$

Assuming that the circuit of multiply as it represented in Figure 4.21 the followings equation will calculate the area and latency of the Multiply circuit.

$$\Delta_{Multiply} = \Delta_{\lambda bitAdd} + \Delta_{\Lambda bitB-Shifter} + \Delta_{2InpXOR} + \Delta_{\Lambda bitAdd} \tag{4.42}$$

$$A_{Multiply} = A_{\lambda bitAdd} + A_{\Lambda bitAdd} + \Lambda \times A_{2InpXOR} + A_{\Lambda bitB-Shifter} + (\Lambda + \lambda) \times A_{2InpMux} \tag{4.43}$$

Considering the circuit represented in Figure 4.21 for Scaling, we calculate the delay and area requirement of this circuit for a 32 bit representation.

$$\begin{aligned}
\Delta_{Scale} &= \Delta_{32bitB_Shifter} + \Delta_{32bitSub} \\
&+ \Delta_{2InpMux} + \Delta_{32bitAdd} + \Delta_{2InpMux} \\
&+ \Delta_{32bitAdd} + \Delta_{2InpMux} + \Delta_{32bitAdd} \\
&= \Delta_{32bitB-Shifter} + 4 \times \Delta_{32bitAdd} + \Delta_{Inv} \\
&+ \Delta_{2InpMux} + 2_{4InpMux} \\
&= \Delta_{32bitB-Shifter} + 4 \times \Delta_{32bitAdd} \\
&+ \Delta_{Inv} + 5 \times \Delta_{2InpMux}
\end{aligned} \tag{4.44}$$

$$\begin{aligned}
A_{Scale} &= A_{32bitB_Shifter} + A_{32bitSub} \\
&+ 32 \times A_{2InpMux} + A_{32bitAdd} + 32 \times A_{4InpMux} \\
&+ A_{32bitAdd} + 32 \times A_{4InpMux} + A_{32bitAdd} \\
&= A_{32bitB-Shifter} + 4 \times A_{32bitAdd} + 32 \times A_{Inv} \\
&+ 32 \times A_{2InpMux} + 64 \times A_{4InpMux} \\
&= A_{32bitB-Shifter} + 4 \times A_{32bitAdd} \\
&+ 32 \times A_{Inv} + 224 \times A_{2InpMux}
\end{aligned} \tag{4.45}$$

4.5 Results

We implemented the architecture presented in this work. Our implementation uses a 16 bit fixed point representation at the input (this number changes in the internal levels). We did not consider use of pipelining technique since pipelining and its achievable gain is orthogonal to the main idea of this work. The use of the pipelining and parallel hardware can be employed without any complications since the matrices are independent of each other. The use of parallel hardware can increase the throughput while it dose not effectively benefit the hardware efficiency and our design hardware efficiency will be poorer than some of the previous designs; on the other hand using pipelining will increase the throughput as well as hardware efficiency.

The comparison of the presented method in this with some of the state of the art works is presented in inverse chronological order in Table 4.3. This results show considerable improvement in energy per matrix target function over other designs. The sate of the art designs apply their algorithm on complex matrices, while our approach is employed to take advantage of reduced complexity achievable. This approach is valid since the recent publications are using SVD on the complex valued channel matrix of telecommunication systems; however for an application that requires the decomposition of a real valued matrix it is not that efficient. Any complex valued channel matrix can be converted to real valued channel matrices with four times the number of elements. To keep comparability we presented our synthesis results in a comparable form. A 2×2 matrix size in Table 4.3

is a 2×2 complex valued matrix and equivalent to a 4×4 real valued matrix.

For comparison we considered the most resent designs that are able to calculate the SVD of nonsymmetric matrices and decompose a matrix to three matrices of U, Σ, *and* V. Our goal is to show how our proposed design is able to provide energy efficient design with minimal hardware complexity. Due to the lack of pipelining and parallelism our proposed hardware does not achieve high hardware efficiency but the achievable gain from those methods is orthogonal to the benefit of this work original idea. In Table 4.3 the results for various target functions are presented. In the telecommunication era power consumed to accomplish a task (as an indicator to show how fast the portable devices will drain their power using that particular device), and the throughput of a system are of the most importance. The value of the target function energy per matrix holds the effect of both important parameters. This parameter does not include the effect of hardware complexity. energy per matrix is the target function that can be used for comparison in this case. This function is able to project the effect of power consumption and throughput at the same time and ignores the effect of orthogonal techniques used for reducing the power, however, this function is still not able to eliminate the effect of hardware reuse in pipelining.

Beside the works compared in the Table 4.3 authors in [56] use a method called supper linear SVD (SL-SVD) and are able to decompose a matrix sizes of $1 \times 1 \sim 4 \times 4$ very efficiently. The only downside to this algorithm is that it relies on the matrix quality and the channel characteristic. This matrix characteristic is harder to achieve with bigger size matrices. The convergence of this algorithm riles on all the singular values being different

and in the case of two or more singular values being equal this algorithm never converges.

In comparison with [57] our design achieves a lower throughput in smaller matrix sizes while our design without using any pipelining is able to achieve 23% higher throughput for an 8 channel matrix. Authors in [57] use different bit sizes ($12 \sim 16$) for different matrix size ($2 \times 2 \sim 8 \times 8$) and uses different number of sweeps ($3 \sim 15$) for various matrix sizes. This method is also only designed to calculate the SVD of square matrices with even number of rows with complex elements. Our design on the other hand keeps the 16 bit[5] input accuracy for all the matrix sizes, and maps the number of sweeps used in [57] to the equivalent number for fast rotations method based on the values demonstrated in Figure 4.10. The proposed design is also able to decompose any square matrix of size $1 \times 1 \sim 8 \times 8$ (complex valued elements) and is able to as effectively decompose matrices with real valued element. In term of energy efficiency or our target function of energy per matrix our design provides $2.83 \sim 5.32$ (2.83 achieves from comparison of 8×8 matrices and 5.32 from comparison of 2×2 matrices) times better efficiency.

The authors of [58] and [59] do not provide the power consumption of their design. The hardware complexity of the design represented in these works is considerably lower specially for lower sized matrices, however the throughput and normalized throughput of those designs is also lower. This in other word means that the hardware efficiency of these designs is lower and they are not good candidates for high throughput applications.

[5] To achieve the required accuracy in 16 bit implementation in scaling circuit it is only critical to implement the B-Shifter, Subtractor, and first adder.

The authors in [60] use a Givens Rotation based design with a bipartite decomposition algorithm. First they convert a general matrix to a bidiagonalized matrix. The next step is to nullify the off diagonal elements. This design is capable of calculating both SVD and QRD (QR Decomposition). The proposed architecture in [60] is only capable of decomposing 4×4 matrices. Various techniques including pipelining, hardware sharing, and early termination are utilized to increase the hardware efficiency and throughput of the design. In the Table 4.3 the power consumption of the design is mentioned with and without utilization of early termination process. this is to provide a fair comparison number since the gain achieved from early termination is application specific and also other designs could employ it. This design adopts a 12 bit implementation in contrast with our 16 bit implementation. The energy per matrix function of this design is 15.4% better than our proposed design and its hardware complexity is 0.72% higher.

In comparison with [61] the energy per matrix of our proposed design is $649 \sim 36.131$ times for matrix sizes of $2 \times 2 \sim 8 \times 8$ respectively. This benefit is achieved as the result of lower power usage of our work which is due to the simplicity of the hardware and eliminating the need for multiple pipeline registers. The achievable throughput and normalized achievable throughput are also higher in our proposed design.

In conclusion in term of Throughput as the target function, this design shows a superior performance compared to the works presented in [58], [61], and [59]. The work in [57] is 4.51 times better than our design for matrix size of 2×2; while this gap in the throughput result is reduced with the increase in the matrix size, and our design provides 1.24 times

better throughput for 8×8 matrices. This achievement is considerable since our design does not use any parallelism or pipelining. In term of hardware efficiency or Normalized Throughput function the design presented in this work provides better results than the work presented in [58], [61], and [59]. While the normalized throughput of this work is $1.47 \sim 3.931$ times for matrix sizes of $2 \times 2 \sim 8 \times 8$ respectively compared to the work presented in [57], this is not far from expected since we expected the hardware efficiency of our design to be poorer.

4.6 Conclusion

In this work, for the first time we presented an algorithm that directly estimates the fast rotations for singular value decomposition of a non symmetric matrix. This method, unlike the previous efforts to implement an eigenvalue decomposition, is able to provide the "Normalized" results. An implementation is presented for 2×2 matrix as the basic block cell of any matrix of higher size. Unlike the previous efforts the proposed hardware does not require any floating point representation or hardware. The hardware complexity of the 2×2 matrix decomposer is much lower than the previous floating-point implementations. This design provides $2.83 \sim 649$ times better energy per matrix performance compared to the most resent designs. The benefit achieved is all due to the proposed algorithm and hardware and other orthogonal hardware implementation methods (pipelining, hardware reuse, and parallel computing) can be applied to increase the hardware performance.

Table 4.3: Comparison of different decomposition algorithms.

	This Work	[57]	[58]	[60]	[61]	[59]
Technology (nm)	90	90	90	90	90	90
Algorithm	ERFHSVD	napSVD	2-Sided Jacobi	GR Based	Adaptive SVD	GK
Functional	U, Σ, and V	U, Σ, and V	U, Σ, and V	U, Σ, and V	U, Σ, and V	U, Σ, and V/Q, R, and P
Matrix Size	$1\times1 \sim 8\times8$[1]	$2\times2/4\times4/6\times6/8\times8$	$2\times2/4\times4/6\times6/8\times8$	4×4	$1\times1/2\times2/3\times3/4\times4$	$4\times4/8\times8/16\times16$
Max. Frequency (MHz)	125	752	112	143	101.2	400
Gate Count (KGE)	116.7/448.3/$1756.1	359	378	451.53	543.9	54.5
Power (mW)	16.8/64.5/$252.8	402/595/673/770	-	164.4/218.29[2]	125	-
Throughput (MMatrices/s)	41.7/8.93/$2.08	188.1/15.7/6.3/1.68	-/0.229/-/0.09866	35.75	0.4791	0.01391/0.00136/-
Normalized Throughput (Matrices/s/GE)	357/19.92/$1.19	524/43.7/17.5/4.68	-/0.607/-/0.261	7.87	0.88	0.255/0.025/-
Energy per Matrix (nJ)	0.4021/7.222/121.4	2.14/38/107.4/343.6	-	4.6/6.11[2]	260.9	-

[1] Only the results for $2\times2/4\times4/8\times8$ is mentioned in the table in favor of simplifying the presentation.

[2] The power numbers are considered with/without early termination method.

5 INTEGER FORCING PROCESSOR BASED ON SLOWEST DESCENT METHOD

In this work we present the hardware implementation of Integer Forcing detector. This design uses singular value decomposition (SVD) for the calculation of Eigen values, Eigen Vectors, and matrix pseudo inverse. Since SVD is used as the basic building block for all these functions we developed a separate processor to calculate SVD in parallel with other tasks. The proposed architecture extends the capabilities of SVD hardware evolves it to an Integer Forcing (IF) decoder detector. This detector is implemented in 90 nm CMOS technology. We compare the performance and hardware implementation cost of this detector with two state of the art detectors (a linear and a close to maximum likelihood detector). The implementation results are inline with our software defined radio complexity analysis and show considerable advantage in terms of performance, compared to MMSE detector (13.5 dB); and in term of hardware complexity, compared to close to maximum likelihood detector (8 times).

5.1 Introduction

The advancement of telecommunication technologies is unavoidable. This advancement demands the use of more transmitter and receiver antennas, higher order in modulation constellation, and computationally more complex algorithms in transmitters. These

demands result in higher computational complexity requirement by the baseband processor and leaves us with a dilemma. The dilemma is to use the limited processing power and energy of a system to achieve higher throughput or increased accuracy (higher quantification resolution, or more iterations in iterative algorithms) in the implementation of transceiver algorithms. We also have a choice between multiple algorithms with varying degrees of accuracy and complexity.

Multiple input multiple output (MIMO) is a method of increasing the data transfer throughput using multiple transmit and receive antennas. MIMO transmission over a channel can be presented as $\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{z}$, where $\mathbf{x}$ is the transmitted signal vector, $\mathbf{H}$ is the channel matrix, $\mathbf{y}$ is the received signal vector, and $\mathbf{z}$ is the noise vector. In general, elements of $\mathbf{y}$, $\mathbf{H}$, $\mathbf{x}$, $\mathbf{z}$ are in $\mathbb{C}$; but without loosing generality we assume real presentation for convenience [34].

In these equations $\mathbf{H}$ is a $L \times N$ matrix where L is twice the number of received and N is twice the number of transmit antennas. Consequently $\mathbf{x}$ is an $N \times 1$ vector and $\mathbf{z}$ and $\mathbf{y}$ are $L \times 1$ vectors. In this work we assume that the receiver knows $\mathbf{H}$, $\mathbf{y}$, and the constellation mapping of $\mathbf{x}$. Also, we assume $L = N = 8$ (equivalent of a 4×4 antenna arrangement),the constellation is 16QAM (quadratic amplitude modulation) with the peak power of P, and noise and channel matrix elements are independent and identically distributed (i.i.d.) random variables with average of zero and variance of $\sqrt{2}/2$ (this makes the variance of complex distribution is equal to one).

Representative algorithms from two general categories of MIMO detector are consid-

ered in this work for comparison purposes. First, search based methods that are more complex category of MIMO detectors. This category of detectors search a subset of possible space to find the transmission with highest possibility. The search has to be performed for each transmission (144 transmission are in a packet with our assumptions). The maximum likelihood (ML) detector in this category is a detector that is able to find the transmission with highest possibility through the whole search space. The ML detector tries to find the minimizing argument in the following equation,

$$\underset{\widehat{\mathbf{x}}}{\operatorname{argmin}} |\mathbf{y} - \mathbf{H}\widehat{\mathbf{x}}|^2, \tag{5.1}$$

where, $|.|$ denotes the absolute function, and $\widehat{\mathbf{x}}$ is every possible combination of constellations. Other types of search based detectors try to minimize the search space and complexity of the search while they are minimally affecting the performance of the system.

The second category is the linear detectors. In these detectors the detection procedure for linear receivers can be modeled as matrix $\mathbf{B}$ multiplied by the received signal vector, while each type of detector defines a unique $\mathbf{B}$ matrix. Then we can assume that the detected signal is,

$$\mathbf{d} = \mathbf{B}\mathbf{y} = \mathbf{B}\mathbf{H}\mathbf{x} + \mathbf{B}\mathbf{z} = \mathbf{A}\mathbf{x} + \mathbf{n} \tag{5.2}$$

while, $\mathbf{A} \equiv \mathbf{B}\mathbf{H}$ and $\mathbf{n} \equiv \mathbf{B}\mathbf{z}$. In general, as the complexity of the algorithm to calculate the $\mathbf{B}$ increases the PER performance of that detection method improves. While the ZF approach nullifies the interference between channels so that $\mathbf{A}_{ZF} = \mathbf{I}_L$ with the cost of increasing the noise power, the MMSE detector aims to maximize the post-detection signal-

to-interference plus noise ratio (SINR). The ZF and MMSE detectors calculate the **B** matrix as follows:

$$\mathbf{B}_{ZF} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T, \tag{5.3}$$

$$\mathbf{B}_{MMSE} = (\mathbf{H}^T\mathbf{H} + \frac{1}{P}\mathbf{I}_L)^{-1}\mathbf{H}^T \tag{5.4}$$

where $(.)^T$ denotes the matrix transpose operation and I_L is the identity matrix of size $L \times L$. Assuming the channel is constant during the transmission of one packet, the complexity of these detectors as well as their performance is much lower.

While the search within a set of candidates for every transmission makes the search based detector expensive, and the performances of linear detectors are generally poor, the integer forcing (IF) detector uses the basic property of lattice codes to find an efficient linear receiver, ideally without increasing the noise. The IF detector recognizes the algebraic structure of lattice codes and its property that an integer combination of a lattice codeword is still a lattice codeword, and tries to find a full rank matrix **A** such that, all the elements of **A** are integer values, and also the achievable rate is maximized. In fact the IF detector allows a wider range for **A** matrix instead of restricting it to identity matrix (in case of ZF detector). The IF detector finds the **A** matrix using a search method but this search has to be done only once for each transmitted packet since we assume the channel matrix is constant during a packet transmission (slow fading channel). Also, using a proper method [4] the search complexity of IF detector can be reduced effectively. After finding the $\mathbf{A}_{IF}$, (5.5) can be used to obtain the $\mathbf{B}_{IF}$.

$$\mathbf{B}_{IF} = \mathbf{A}_{IF}\mathbf{H}^T(\mathbf{H}\mathbf{H}^T + \frac{1}{P}\mathbf{I}_L)^{-1} \tag{5.5}$$

The rest of this work is organized as follows.

5.2 Soft Output Integer Forcing Detector

The hard detection of IF detector offers a better performance compared to the hard detection of other linear detectors in return for extra hardware complexity. This extra hardware complexity is neglectable compare to search based detectors. However, there are challenges in using IF detectors that we try to address them before starting the hardware implementation. The first problem is the soft detection and log likelihood ratio LLR calculation of the received symbols in this method. The other challenge in implementing a receiver with IF detector is that the current standards do not support a consolation of prime field. The third challenge is finding a method that can effectively construct the $\mathbf{B_{IF}}$. In [62] a solution for the hard detection of QAM constellations demapping is proposed.

In this section first we overview the algorithm presented in [4] and the changes we need to apply to this algorithm, this algorithm help us to effectively generate the receiver matrix. Then we extend the hard detection method presented in [62] to two stage soft detection, and show that its performance is far superior to MMSE detectors with soft decoding.

5.2.1 Modified Slowest Descent method

The reference [4] presents two algorithm that generate $\mathbf{A}_{IF}$ together. The Algorithm 7 takes $\mathbf{Q}$ [1], sorted Eigen vectors [2], J[3], and M[4] and generates a set of candidates Ω[5]. Using sorted ω and $\mathbf{Q}$, Algorithm 8 constructs a full rank matrix $\mathbf{A}_{IF}$ such that the total achievable rate is maximized. We slightly modified this algorithms from [4] for the reasons that will follow.

We decided to ignore section *(iv)* of Step 2 in Algorithm 7. There is no reason to clear the points not in range. Ignoring this instruction will reduce the computational complexity and increase the chance of achieving the close to optimum solution with smaller M. In fact the original algorithm requires atleast $J = 5$ and $M = 5$ for optimum solution while the same result is achievable if we ignore this instruction $J = 5$ and $M = 3$. We only had to limit the boundaries of this step due to fix point requirements $[-8, 7]$. So, we choose $M = 3$ for Step 1 and limit the results of Step 2 to $[-8, 7]$.

In Algorithm 8, the original algorithm does not provide any solution for case that the number of candidates are less than L and the full rank $\mathbf{A}_{IF}$ is not achievable. Unsuccessful construction of $\mathbf{B}_{IF}$ has severe effect on BER performance since it causes a burst of errors. This will result in loss of a complete packet which will affect the BER (bit error rate) performance of the system more than the normal effect it would have on PER (packet

[1] $Q \equiv A_{IF} H^T (HH^T + \frac{1}{P} I_L)^{-1}$
[2] $\mathbf{g_1}, \mathbf{g_2}, ..., \mathbf{g_L}$ of Matrix $\mathbf{Q}$ with corresponding Eigen values $\lambda_1 \leqslant \lambda_2 \leqslant ... \leqslant \lambda_L$
[3] Number of slowest descent lines.
[4] Bound for each coordinate of searching vector.
[5] Candidate set.

error rate) performance. In another word, using the original algorithm even in very high Eb/No the BER (also PER when boundaries are narrow) will have an error floor due to the lost packets (an unsuccessful construction of $\mathbf{B_{IF}}$ is additional error to the error caused by channel noise, and it should be considerably less than 1% or 0.1% if low PERs are target. In that case the probability of an unsuccessfully $\mathbf{B}_{IF}$ contraction should be zero which highlights the importance of change we made in Step 3 of Algorithm 7. The $abs()$, and $diag()$ functions are the functions that accumulate all the input elements and extract the diagonal elements of a matrix respectively. The function $n()$ represents size of the elements in a set. In this case $\mathbf{A}_{MMSE}$ is the identity matrix of size 8×8.

Algorithm 7 Candidate Set Search Algorithm with Slowest-Descent Method

Input: $\mathbf{Q}$, M, $(1 \leqslant J \leqslant L-1)$, $\mathbf{g_1}, \mathbf{g_2}, ..., \mathbf{g_L}$.
Output: Search vector candidate set Ω.
Step 1: Compute the set of Λ:

$$\Lambda = \{m_j, \quad j = 1,2,...,2M+2\}$$
$$= \{-M - \frac{3}{2} + j, \quad j = 1,2,...,2M+2\}$$

Step 2: For $i = 2,...,J+1$,
(i) Obtain the jump points where:

$$\rho\prime_{(j-1)\times L+k} = \frac{m_j - g_{1k}}{g_{ik}}$$

with $k = 1,...,L, \quad j = 1,2,...,2M+2$.
(ii) Sort $\rho\prime_1, ..., \rho\prime_{(2M+2)\times L}$ to $\rho_1 \leqslant ... \leqslant \rho_T$
(iii) Find the closest point $\mathbf{a}_{(\rho_j, \rho_{j+1}],i}$ as:

$$\mathbf{a}_{(\rho_j,\rho_{j+1}],i} = round\left(\mathbf{g}_1 + \frac{1}{2}(\rho_j + \rho_{j+1})\mathbf{g_i}\right)$$

with $(\rho_j, \rho_{j+1}], i = 1,...,T-1$
(iv) Clear the points that are not in range $[-M, M]$ in any coordinate, exclude the zero vectors, and the remaining vectors in Ω_{i-1}.
Step 3: Make $\Omega = \Omega_1 \cup ... \cup \Omega_J$.

Algorithm 8 New Search Algorithm to Form $\mathbf{A_{IF}}$.

Input: $\mathbf{Q}$, Ω.
Output: Full rank matrix $\mathbf{A_{IF}}$.
Step 1: Let $f(\mathbf{t}) = \mathbf{t^T Q t}$ and sort all vectors in the searching candidate set Ω such that:

$$\Omega = \{\mathbf{t}_1, ..., \mathbf{t}_{n(\Omega)} : f(\mathbf{t}_1) \leqslant ... \leqslant f(\mathbf{t}_{n(\Omega)})\}$$

and set the $\mathbf{a_1} = \mathbf{t_1}$, initialize $i = 1$, and $j = 1$.
Step 2: Form $\mathbf{G}$ from the first L vectors of Ω, and calculate $\mathbf{R}$=qr($\mathbf{G}$), $i = 8$, j=sum(diag($|\mathbf{R}| > 0$))
While $i < n(\Omega)$ and $j < L$, do
(i) Remove $L - j$ vectors from the $\mathbf{G}$ that their associated diagonal element is zero (diag($|\mathbf{R}|$=0))
(ii) Append the $L - j$ next vectors of Ω to $\mathbf{G}$
(iii) $i = i + L - j$, $\mathbf{R}$=qr($\mathbf{G}$), j=sum(diag($|\mathbf{R}| > 0$))
Step 3: If $j = L$, then $\mathbf{A}_{IF} = \mathbf{G}$, else $\mathbf{A_{IF}} = \mathbf{A_{MMSE}}$.

5.2.2 Detection Method

The detection process has to be completed with demodulation after multiplication of $\mathbf{B}_{IF}$ and received signal. The hard detection process is completed with finding the closest possible constellation to the received signal and then demapping the constellation to data. If the constellation is of lattice code in a closed field of prime order then any integer combination of lattice codes will be a lattice code of the field. However, the standard uses QAM constellation mapping and does not hold this property. This turns the detection to a search problem within the unique set of possibilities for each $\mathbf{B}_{IF}$ (upto CN^L, and CN is the number of constellations in the QAM.

A hard detection method that address this is offered in [62]. This solution instead of using the properties of a close field argues that since the elements of $\mathbf{A}_{IF}$ are integer and also the constellations are integer numbers, then after the multiplication the result should be an integer. Using this property we can find the closest integer to any of the received constellations and avoid the need for the search. We should mention that, this is not an exact solution and not every integer is in the real of possibilities for each detected constellation.

In fact, only a limited set of integers have to be considered to achieve the exact solution, however this will make the detector a nonlinear detector and that won't be acceptable.

In [63] the performance comparison of IF detector with ML, ZF, and MMSE detector is presented. The performance curves are for hard detection of the received signal with different detection algorithm. As expected the performance of IF hard detected signal outperforms the hard output MMSE detector; however, the soft information with MMSE detectors are easily obtainable and outperforms the IF hard detector. This is while obtaining soft information in IF detector is not as straight forward and for ML detectors the soft information achieved through calculating a posterior probability (ML-APP) within the list of candidates. So, the dilemma is whether we implement an MMSE detector with soft decoder or an IF detector with hard decoder or find a third option.

Modern communication systems require high-performing FEC engines, and soft decoding LDPC is the natural choice [64]. This motivates us to find a method that the soft information can be extracted from IF detected signals. In the IF detector the detection happens in a phase that the received symbol and their locations in the constellation map is not representative of the data they are carrying. This fact makes the soft detection of signals challenging.

While the methd suggested in [62] enables us to detect the signals, we need to extend it to achieve soft information. In order to use this method, the $\bar{Y}$ should be calculated as

follows:

$$\bar{\mathbf{y}} = \mathbf{y} + \mathbf{Hc} \tag{5.6}$$

$$\bar{\mathbf{d}} = \frac{\rho}{2}(\mathbf{B_{IF}}\bar{\mathbf{y}}) \tag{5.7}$$

$$\mathbf{d}' = Round(\bar{\mathbf{d}}) \tag{5.8}$$

$$\hat{\mathbf{d}} = \mathbf{A_{IF}^{-1}}\mathbf{d}' - \mathbf{s} \tag{5.9}$$

Where $\mathbf{c}$, is an $L \times 1$ vector with each element equal to $C = \frac{log_2^{CN}-1}{\rho}$, and $\mathbf{s}$, is an $L \times 1$ vector with each element equal to $S = \frac{\sqrt{CN}-1}{2}$. Also, ρ is the normalizing factor ($\sqrt{2}$ for QPSK, $\sqrt{10}$ for 16 QAM, and $\sqrt{42}$ for 64 QAM) which normalizes the power of the mapped constellation to one. The final step is to feed $\hat{\mathbf{d}}$ to a soft demodulator witch uses the approximate LLR (approximate likely hood is calculated based on the distance of received symbol from its nearest constellation instead of the reserved symbol general location or how it is located in comparison to all the constellations) calculation method to obtain the soft values. This is demonstrated more clearly in the flowchart that we present in the next subsection.

5.2.3 The IF Receiver Block Diagram

Figure 5.1 shows the block diagram of the IF detector. This block diagram contains the detector and decoder of the receiver. The detector is derided into two sat blocks. The first sub block (on the left hand side of the figure) has to calculate the outputs once for every packet transmitted. The second sub block of the detector (demonstrated o n the right hand side of the figure) has to calculate the outputs based on the provided inputs once per symbol

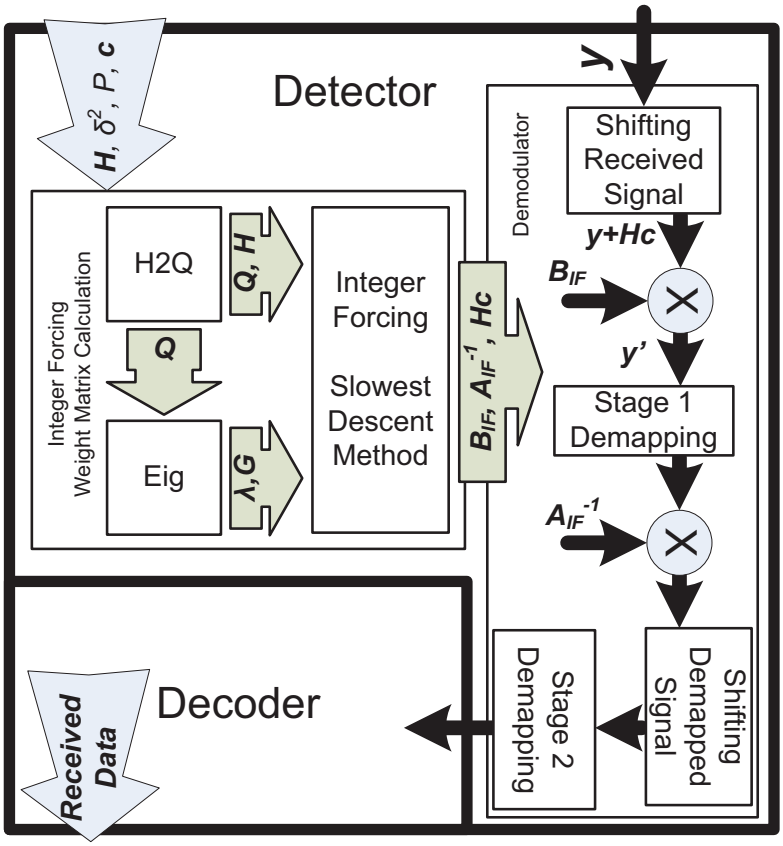


Figure 5.1: The block diagram of IF receiver.

(144 times the other sat block). This sub block mathematically is less complex, but it has to calculate 144 vector by matrix (8×1 and 8×8 in this case) multiplication. To achieve the required throughput our design has to be able to apply this multiplication in one clock cycle.

5.3 Fixed Point and Cycle Accurate Hardware Model

In this section we focus on limiting the resource demanding parameters of the design in order to reduce the complexity of the hardware required to perform the task in hand. This

process contains both limiting the computational accuracy (limited fined point accuracy) and also,limiting the number of iterations and size of every resource demanding task. As the result, a loss in system PER performance is inevitable. Of course, this performance loss is controllable but, 0.5 dB is considered to be reasonable loss in performance due to quantification [65] [66]. Normally this reasonable performance loss is expected at the PER of 10^{-1} to 10^{-2}. The quality of service (QoS) requirements are normally only applied to this range; however, since we are expecting bigger packet size in future standards, the PER range of 10^{-1} to 10^{-3} is considered in this work.

In our effort to find the optimum fixed point model we used truncation to handle the list significant precision bits and overflow happens in case the integer bits are not enough, unless otherwise mentioned. This provision (allowing the overflow to occur instead of using saturation and truncating the precision bits instead of truncation) is taken to reduce the hardware complexity. In case that the overflow happens too often this will cause serious performance loss in the system; however, for each signal we measured the maximum value it achieves over 10^5 frames and made sure overflow does not occur. Occasional overflow is either by decoder or does not have significant effect on performance of the system.

In addition to aforementioned efforts we had to apply necessary changes to algorithm so that it is efficiently implementable in the hardware. The model also had to be able to represent the change in performance caused by particular hardware implementation , as we will discuss in detail. The fix point cycle accurate model is then used to verify our hardware implementation. This will guaranty the correct implementation o the model and

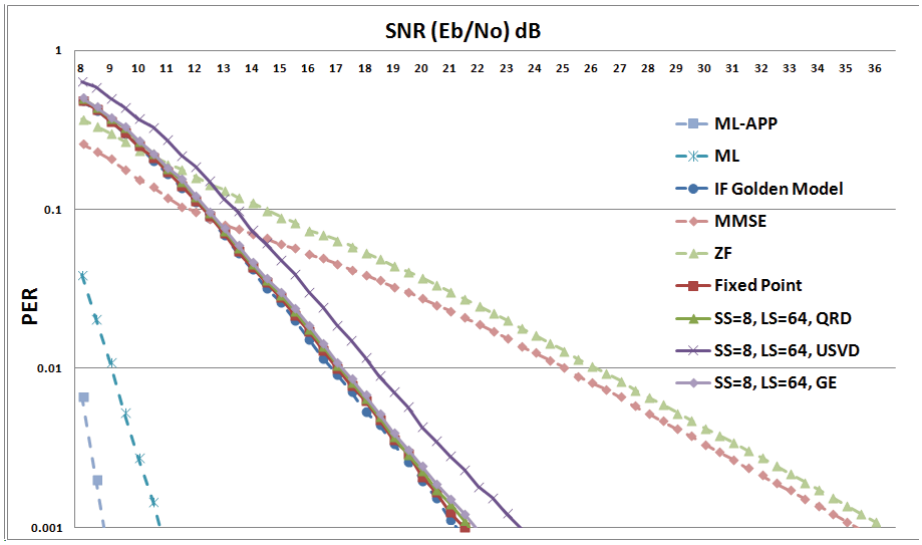


Figure 5.2: Performance for different steps of implementation.

also make sure in similar environment as simulations assume the hardware performance matches our expectations. In Figure 5.2 the dashed lines demonstrate the performance of ML-APP, ML, MMSE, ZF, and IF Golden Models. The golden model is a term when the system is presented with double precision floating point arithmetic. In this model the LDPC encoding rate is $\frac{1}{2}$, and the data transmission packet size is 1154 bits (144 burst of 16 QAM symbols over a 4×4 antenna arrangement).

5.3.1 Quantization Effect

Figure 5.3 to Figure 5.6 show the flowchart of the detector. The detector receives the channel (**H**) matrix and transmitted symbols and provides the soft detected symbols to the decoder. Every signal is named once on the flowchart and the required precision is mentioned in Table 5.1. In addition, Table 5.2 demonstrates the required precision of two pseudo inverse matrix operation [67] used in the detector. The naming of signals is in

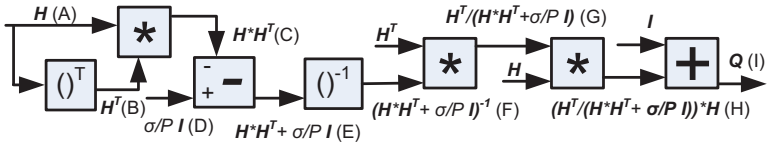


Figure 5.3: Converting channel matrix to the symmetric matrix.

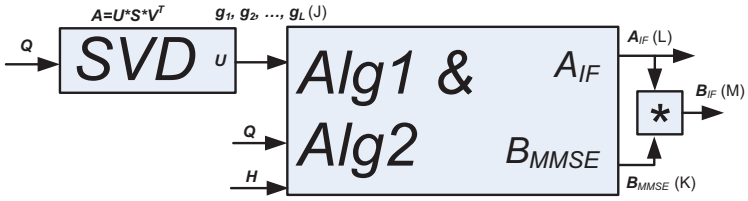


Figure 5.4: Forming the full rank integer matrix.

respect to what is presented in Figure 5.7. Based on our simulation results reducing the number of precision bits to values less than what is mentioned in Table 5.1 and Table 5.2 would severely damage the performance of the system and we would not be able to hold the conditions set at the beginning of this section. The doted curve demonstrates the performance of a system where all the parameters and signals are fixed based on the information mentioned in these tables. The dotted curve in the Figure 5.2 demonstrates the performance of the system after every signal in the system is quantized a cording to these tables information. The solid line curves in the figure represent the performance of the system after further changes to the algorithm and system has been applied as will be discussed in the next subsection.

Table 5.1: Number of bits required by different signals in the detector.

Signal	Precision	Signal	Precision	Signal	Precision	Signal	Precision	Signal	Precision	Signal	Precision	Signal	Precision
A	13SFix10	B	13SFix10	C	19SFix13	D	13SFix10	E	19SFix13	F	28SFix17	G	22SFix17
H	19SFix18	I	19SFix18	J	9SFix8	K	15SFix10	L	4SFix0	M	14SFix10	N	18SFix16
O	15SFix10	P	14SFix9	Q	14SFix9	R	15SFix9	S	15UFix13	T	8SFix1	U	7SFix0
V	3SFix1	W	13SFix3	X	23SFix15	Y	14SFix3	Z	13UFix13	AA	14SFix4	AB	7SFix5

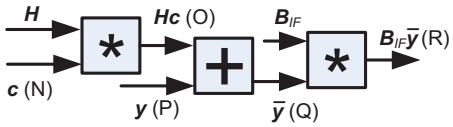


Figure 5.5: Shift in received signals.

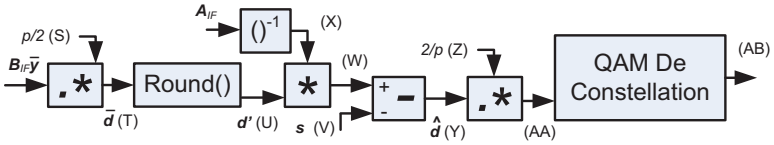


Figure 5.6: Soft demapping of received signals.

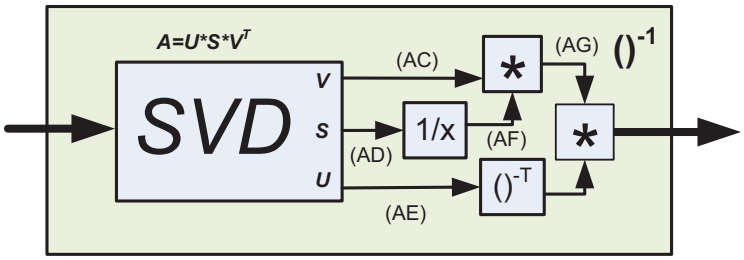


Figure 5.7: Flowchart of the pseudo matrix inversion.

Table 5.2: Number of bits required by different signals in the inverse function.

	In	AC	AD	AE	AF	AG	No. of SVD sweeps	No. of Inv. Itr.	Out
$\mathbf{A_{IF}}^{-1}$	4SFix0	16Sfix15	16Sfix10	16SFix15	26SFix17	20SFix12	7	1	23SFix15
$(\mathbf{H}*\mathbf{H}^T+\mathbf{S}/P\mathbf{I})^{-1}$	19Sfix13	18Sfix17	20SFix14	18Sfix17	29SFix18	32SFix21	8	7	28SFix17

5.3.2 Cycle Accurate Hardware Modeling

We also did applied other changes to the algorithm and hardware model for making them hardware friendly. The first change that we already have mentioned it in the previous subsection was the change of matrix inverse function to pseudo matrix inverse. This change did not had any effect on the performance of the golden system model (when both functions are implemented with ideal parameters. The last three columns of Table 5.2 show the number of SVD sweeps, Number of inverse function iterations (one clock per iteration), and required precision bits at the output of pseudo inverse function. For SVD design we used a design based our previous work [68],and for inverse function we are using a Taylor based solution that we will explain in more details in the next section.

In order to reduce the hardware complexity we investigate limiting the candidate list size in that is the output of Algorithm 7. Ignoring the Algorithm 7 sub-step *(iv)* of **Step 2** has already effectively reduced the maximum required list size (from 480 to 320) but this is not enough. Specially if we notice that we need to calculate the cost associated with each candidate in the list and then sort the candidates based on their cost. We concluded that the list size of 64 (LS=64 in the figure) atleast (we only considered the list sizes of power of two) is required to achieve near optimum performance. Next we investigate the possibility of reducing the required sort complexity.

Explaining our choice of proper sort architecture without considering the design we had in mind is impossible. The idea was to select eight candidate at a time, Calculate their

costs and then put the costs and candidates in a buffer with ability to sort. This provision enables us to run the sort in parallel. We used bobble sort with different sort net work size in order to investigate the effects each might have on the performance of the system. This investigation was contingent on the assumption that four clock cycles are used in the process of calculating the costs each time. Two arrangements were able to provide close to optimum performance. One with the sort network of eight (SS=8) In this case the total cost of sort in Algorithm 8 will be four clock cycles. The downside of this method is that the list size need to be increased to 72. The second option is to use use sort network of size four (SS=4), this requires four clock cycles after each time costs are calculated; however, 11 clock cycles also needed every time the 64 cost values associated with each Eigen vector is calculated. This means the sort process mentioned in Algorithm 8 will cost us 55 clock cycles.

Every time that buffer reaches its full capacity we first calculate the next eight cost values then we open eight spot in the buffer to take full advantage of parallel working sort buffer (we basically overwrite the last eight values with eight new values). The associated vectors to the cost values are compressed and stored in the second layer of Sort buffer and they are rearranged following with the rearrangement of their costs. The vectors are constructed from integer values in range of $[-8, 7]$ and we needed only four bit for their representation and 32 bits in total are required for storing a vector of size eight. This fact enable us to calculate the cost of eight vectors in parallel with slight modification to the hardware we had. We will explain the hardware provisions that enables vector com-

pression, decompression, and parallel cost calculation in the next section. The maximum number of candidates that the system keeps in the sort buffer is different from what is mentioned before only in case that all the cost values associated with 64 vectors of a certain Eigen vector are calculated. In that case we only keep eight candidate per Eigen vector unless it is the last one of the five.

In addition, we limit the number of efforts in forming the $\mathbf{A}_{IF}$ (number of times QRD is performed) to 20. In the Figure 5.2, the green solid line carve with rectangular indicator demonstrate the performance of hardware accurate model in different SNRs when all the aforementioned provisions applied. Our final attempt in reducing the hardware complexity was to replace QRD with SVD for matrix rank calculation and to determine the dependent vectors. We considered this to be the natural decision at this point since the hardware to support SVD already did enlist in our design for the other task such as calculating the Eigen vectors, Eigen values, and pseudo matrix inversion. The rank calculation with SVD is considered to be numerically more stable and the most reliable rank calculation method [69]. The downside to using SVD is its computational complexity; however, our proposed SVD architecture was at he to effectively reduce the required hardware complexity. Replacing the QR decomposition with SVD did not provide the desired result.

The purple solid line with the circular indicator in Figure 5.2 demonstrates the performance of the system in this case. The performance curve in this case shows 2 dB reduction in performance. The maim reason is that with SVD when there is a dependency between two or move vectors we do not have any control on which vector (the one with higher cost

or the one with lower associated cost value) will have a diagonal zero element (indicating that vector needs to be replaced). In the case of QRD we pat the vectors with higher cost at the right hand side of the matrix, and decomposition process starts from the left side of the matrix. In case two vectors or more are dependent in this situation, the column(s) on the. nigh would have zero diagonal element, this will result in having a matrix formed from vectors of lower cost and eventual performance increase. We eventually replaced the QR factorization with Gaussian elimination (GE).

The gray solid line with diamond shape indicator in Figure 5.2 shows the performance of the system when row pivoting GE is used to determine the matrix rank. The performance curve in this case fits within our acceptable boundaries. There are various methods to calculate the GE. The proposed architecture is able to calculate a scalar multiply and add in one clock cycle and using this ability we could calculate the rank of a matrix in seven clock cycle; however, the dynamic range of this method is very high. To eliminate this problem we have to use one cycle of our Taylor series based divider each time we try to zero the lower matrix elements, as the result, 14 clock cycles are required to calculate a matrix rank with this method. We have explained this in detail in the next section. Considering that we depend on multiple iterations to perform divisions in our design, the performance gain of QRD implementation would not justify its cost. To help the clarity of this section discussion we have presented the actual algorithm that our design implements in Algorithm 9. The inputs are $\mathbf{Q}$, $\Lambda = \{m_j, \quad j = 1,2,...,2M+2\} = \{-3.5,-2.5,-1.5,-0.5,0.5,1.5,2.5,3.5\}$ and the output is full rank matrix $\mathbf{A_{IF}}$. The

function $GE()$ represents the Gaussian elimination process. Finally it worth to mention that in Algorithm 9 part *(ii)* of **Step 7** and **Step 5**- **Step 6** are processed in parallel.

5.4 Architecture and Microarchitecture of the Processor

This design is divided to three major processing elements. The channel processing, the data processing, and SVD processing. This partitioning is based on the processing power these tasks require in each packet transmission. The channel processing has to processed once for every transmitted packet. We consider a design that is able to perform a vector by matrix multiplication or 64 real number multiply and add/round. We study the complexity of different tasks in each process by element. The SVD is a part of channel processing; however, this element is a demanding task and used as the basic building block of multiple functions. The data processing demands less computational complexity; while,it needs to be able to process the data 144 times faster.

Due to the fact that each packet of data holds 144 symbols, 360 clock cycles are needed to carry out all the task required by channel processing element. Each SVD sweep require seven iterations and assuming each iteration is done in one clock cycle the SVD task needs 133 clock cycles (total of 19 sweeps). Implementing the **Step 1** to **Step 7** in Algorithm 9 require 50 clock cycles and since it needs to be repeated for five times if require the total of 250 clock cycles. This is assuming the sort is done in parallel when possible. The number of pass through the **Step 8** to **Step 10** in Algorithm 9 is unknown. This is the only part of the process that does not have a constant number of ventures. This process takes 62.24 clock

Algorithm 9 Form $\mathbf{A}_{IF}$ Matrix with Slowest-Descent Method

Input: $\mathbf{Q}$, Λ.
Output: Full rank matrix $\mathbf{A_{IF}}$.

for $i = 2,...,6$, **do**
  Step 1: Calculate $\frac{1}{g_{ik}}$
  Step 2: Obtain the jump points where:

$$\rho\prime_{(j-1)\times 8+k} = \frac{m_j - g_{1k}}{g_{ik}}$$

$$\text{with } k = 1,...,8, \quad j = 1,2,...,8.$$

  Step 3: Sort $\rho\prime_1,...,\rho\prime_{64}$ to $\rho_1 \leqslant ... \leqslant \rho_{64}$
  Step 4: Find the mid point of the ranges:)

$$\xi_j = \begin{cases} \frac{1}{2}(\rho_j + \rho_{j+1}) & \text{if } j = 1,2,3,...,63 \\ 3 & \text{if } j = 64 \end{cases}$$

  for $l = 1,...,8$, **do**
    Step 5: Find the closest point $\mathbf{a}_{(l-1)\times 8+j,i}$ as:

$$\mathbf{a}_{(l-1)\times 8+j,i} = round\,(\mathbf{g}_1 + \xi_l \mathbf{g}_i)$$

$$\text{with } j = 1,2,3,...,8.$$

    Step 6: Calculate the cost associate with each vector.

$$f(\mathbf{a}_{(l-1)\times 8+j,i}) = f(\mathbf{a}_{t,i}) = f_{t,i} = \mathbf{a}_{t,i}^T \times \mathbf{Q} \times \mathbf{a}_{t,i}$$

$$\Omega_{i,l} = \{(\mathbf{a}_{t,i}, f_{t,i}) : t = (l-1)\times 8 + j\}$$

$$\text{with } j = 1,2,3,...,8.$$

    Step 7: Manage the sort buffer
    (i) Add the $\Omega_{i,l}$ sets to to Sort buffer

$$\Omega = \begin{cases} \Omega \cup \Omega_{i,l} & \text{if } n(\Omega) < 72 \\ \{\Omega - \{\omega_{65},...,\omega_{72}\}\} \cup \Omega_{i,l} & \text{if } n(\Omega) \geq 72 \end{cases}$$

    (ii) Sort the pairs in Ω with the ascending cost values.
  end for
  Step 8: Keep candidate with lower cost values.

$$\Omega = \begin{cases} \Omega & \text{if } i = 6 \\ \Omega - \{\omega_{8\times(i-1)},...,\omega_{72}\} & \text{if } i \neq 6 \end{cases}$$

end for
Step 9: Initialization
(i) Form $\mathbf{G}$ from the first eight vectors of Ω
(ii) Calculate $\mathbf{R}$=GE($\mathbf{G}$), $i = 8$, j=sum(diag($|\mathbf{R}| > 0$))
(iii) Set $k = 1$ as the counter of number of effort to form $\mathbf{G}$
Step 10: Find the full rank $\mathbf{G}$
while $i < n(\Omega)$ and $j < 8$ and $k \leq 20$ **do**, do
  (i) Remove $L - j$ vectors from the $\mathbf{G}$ that diag($|\mathbf{R}|$)=0
  (ii) Append the $L - j$ next vectors of Ω to $\mathbf{G}$
  (iii) $i = i + L - j$, $\mathbf{R}$=GE($\mathbf{G}$), j=sum(diag($|\mathbf{R}| > 0$))
end while
Step 11: Generate the $\mathbf{A}_{IF}$ output matrix

$$\mathbf{A}_{IF} = \begin{cases} \mathbf{G} & \text{if } j = L \\ \mathbf{A}_{MMSE} & \text{else} \end{cases}$$

cycles by average and 160 cycles in worse case scenario. The total of 624 clock cycles are required in worse case scenario for carrying out the channel processing task. Due to the special architecture of the design and algorithm, two channel processor can share their sort component. Up to four processor can share their sort component with some provisions if joint channel and data a processor is implemented.

Since the singular value decomposition is used for calculating the Eigen values and Eigen vectors of matrix **Q**, and calculating the two inverse matrices (using pseudo inverse matrix calculation). The efficient implementation of this basic block cell is essential. For this element we are using one of our previous designs based on the Givens Fast rotations [68]. This design is based on BLV [5] parallel design with multiple co-processors. Our design benefits from a simplified hardware that is able to sort the Eigen values and corresponding Eigen vectors for a non symmetric matrices. Each sweep of the design takes seven clock cycles. In this work we follow an idea to intend the capabilities of SVD hardware so it can carry out all the tasks designated to channel and data processor. In the went two subsections we first briefly review the design of the SVD processor ant then we explain how we extend to the existing design to achieve the IF processor.

5.4.1 Base Architecture

Fast rotation based singular value decomposition using approximate arithmetic. This algorithm enable us to directly calculate the fast rotations for non-symmetric matrices based on FHSVD algorithm. Assuming a 2×2 matrix is defined as in equation (5.10), FHSVD

algorithm uses (5.11) and (5.12) to calculate α and β; then using those two values and equation (5.13) and (5.14), it calculates the vertical and horizontal Givens rotations. Note that for traditional rotation algorithm (proposed by Gotze [6]), the value of β is zero, and we only need to calculate the value of α (Gotze algorithm is for symmetric matrices only). The calculated rotations then applied to the matrix using equation (5.15). To update the values of **U**, and **V** applying only one rotation is required. Figure 5.8 demonstrates the hardware required to update **U**, and **V** as single Given's rotation hardware.

$$\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \mathbf{U} \times \mathbf{S} \times \mathbf{V}^T \tag{5.10}$$

$$\alpha = tan^{-1}(\frac{c+b}{d-a}) \tag{5.11}$$

$$\beta = tan^{-1}(\frac{c-b}{d+a}) \tag{5.12}$$

$$\Theta = \frac{\alpha+\beta}{2} \tag{5.13}$$

$$\theta = \frac{\alpha-\beta}{2} \tag{5.14}$$

$$\mathbf{S} = \mathbf{R}_{\theta}^{T} \times \mathbf{A} \times \mathbf{R}_{\Theta} = \begin{bmatrix} d_1 & 0 \\ 0 & d_2 \end{bmatrix}$$
$$= \begin{bmatrix} cos(\theta) & sin(\theta) \\ -sin(\theta) & cos(\theta) \end{bmatrix}^{T} \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} cos(\Theta) & sin(\Theta) \\ -sin(\Theta) & cos(\Theta) \end{bmatrix} \tag{5.15}$$

While Gotze method [6] is using the same flow as in the calculation of accurate rotation angles and then divides the results by two to take benefit of double rotations, we use $tan(x) \approx x$ and the fact that this approximation is more accurate for smaller angels ($lim_{x \longrightarrow 0}^{tan(x)} = x$), to provide more accurate estimation of rotation angels. Algorithm 10 explains the necessary steps involved in calculating the rotation angles in a 2×2 matrix. To extend the algorithm to a design of matrix size 8×8 we need to use a design similar to the architecture presented in Figure 5.9. In this design DP represents the diagonal processor and NDP represents a non-diagonal processor. The DPs calculate and apply (using the circuit presented in Figure 5.8) rotations based o n the Algorithm 10. In addition, DPs send out the horizontal and vertical rotations through their available horizontal and vertical bus to NDPs in the same rows and columns,respectively. The NDPs receive the rotation angles via the bus and apply the rotations to the designated elements of **U**, **S**, and **V**. The NDPs only need the circuit in Figure5.8 since they receive the Notation angles and the do not require to calculate it.

Algorithm 10 Algorithm to Directly Calculate the fast Rotation Angles for Non-Symmetric Matrices.

Input: **A**.
Output: Rotation Matrix $\mathbf{R}_{\tilde{\theta}}$ and $\mathbf{R}_{\tilde{\Theta}}$.
Step 1: Calculate the initial values:

$$S_{N1} = Sign(c+b)$$
$$S_{D1} = Sign(d-a)$$
$$N1 = |c+b|$$
$$D1 = 2\times|d-a|$$
$$K1 = exp_2(D1) - exp_2(N1)$$
$$S_{N2} = Sign(c-b)$$
$$S_{D2} = Sign(d+a)$$
$$N2 = |c+b|$$
$$D2 = 2\times|d-a|$$
$$K2 = exp_2(D2) - exp_2(N2)$$

Step 2: Calculate l_α and l_β using following case statement:

$$(l1_{temp}, B) = \begin{cases} (K1+1,1) & \text{if } 1.5\times D1 > 2^{K1+1}\times N1 \\ (K1-1,0) & \text{if } 1.5\times D1 < 2^{K1}\times N1 \\ (K1,1) & \text{if } D1 < 2^{K1}\times N1 \\ (K1,0) & \text{default} \end{cases}$$

$$l_\alpha = max(l1_{temp}+1,2)$$

$$(l2_{temp}, b) = \begin{cases} (K2+1,1) & \text{if } 1.5\times D2 > 2^{K2+1}\times N1 \\ (K2-1,0) & \text{if } 1.5\times D2 < 2^{K2}\times N1 \\ (K2,1) & \text{if } D2 < 2^{K2}\times N1 \\ (K2,0) & \text{default} \end{cases}$$

$$l_\beta = max(l2_{temp}+1,2)$$

Step 3: Calculate the l_θ and l_Θ using the following case statement:

$$(l_\Theta, l_\theta) = \begin{cases} (l_\alpha, l_\alpha) & \text{if } (N2=0) \\ (l_\beta, l_\beta) & \text{if } (N1=0) \\ (l_\beta - (B\&b), l_\alpha) & \text{if } (l_\beta - l_\alpha = -1) \\ (l_\alpha - (B\&b), l_\beta) & \text{if } (l_\beta - l_\alpha = 1) \\ (l_\beta - 1, 0) & \text{if } (l_\beta - l_\alpha = 0) \\ (min(l_\alpha, l_\beta), min(l_\alpha, l_\beta)) & \text{default} \end{cases}$$

$$(l_\Theta, l_\theta) = \begin{cases} (l_\theta, l_\Theta) & \text{if } (S_{D2} * S_{N2} \neq S_{D1} * S_{N1}) \\ (l_\Theta, l_\theta) & \text{default} \end{cases}$$

Step 4: Calculate the $S_{\tilde{\theta}}$ and $S_{\tilde{\Theta}}$ using the following case statement:

$$S = \begin{cases} S_{D1} * S_{N1} & \text{if } (l_\beta - l_\alpha \geqslant 0 \;||\; N2 = 0) \\ S_{D2} * S_{N2} & \text{default} \end{cases}$$

$$(S_{\tilde{\Theta}}, S_{\tilde{\theta}}) = \begin{cases} (S, S) & \text{if } (N2=0) \\ (S, -S) & \text{if } (N1=0) \\ (S * Sign(l_\beta - l_\alpha), S) & \text{default} \end{cases}$$

Step 5: Calculate the $(c_{\tilde{\theta}}, s_{\tilde{\theta}})$ and $(c_{\tilde{\Theta}}, s_{\tilde{\Theta}})$ pairs using the following case statement:

$$\tilde{t_1} = \begin{cases} 0 & \text{if } l_\theta = 0 \\ 2^{-l_\theta} & \text{default} \end{cases}$$

$$\tilde{t_2} = \begin{cases} 0 & \text{if } l_\Theta = 0 \\ 2^{-l_\Theta} & \text{default} \end{cases}$$

$$(c_{\tilde{\theta}}, s_{\tilde{\theta}}) = \frac{1}{1+\tilde{t_1}^2} \times (1-\tilde{t_1}^2, 2\times S_{\tilde{\theta}} \times \tilde{t_1})$$

$$(c_{\tilde{\Theta}}, s_{\tilde{\Theta}}) = \frac{1}{1+\tilde{t_2}^2} \times (1-\tilde{t_2}^2, 2\times S_{\tilde{\Theta}} \times \tilde{t_2})$$

Step 6: Calculate the rotation matrices $\mathbf{R}_{\tilde{\theta}}$ and $\mathbf{R}_{\tilde{\Theta}}$ using the following case statement:

$$\mathbf{R}_{\tilde{\theta}} = \begin{cases} \begin{bmatrix} c_{\tilde{\theta}} & s_{\tilde{\theta}} \\ -s_{\tilde{\theta}} & c_{\tilde{\theta}} \end{bmatrix} & \text{if } S_{D1} < 0 \\ \begin{bmatrix} s_{\tilde{\theta}} & c_{\tilde{\theta}} \\ c_{\tilde{\theta}} & -s_{\tilde{\theta}} \end{bmatrix} & \text{else} \end{cases}$$

$$\mathbf{R}_{\tilde{\Theta}} = \begin{cases} \begin{bmatrix} c_{\tilde{\Theta}} & s_{\tilde{\Theta}} \\ -s_{\tilde{\Theta}} & c_{\tilde{\Theta}} \end{bmatrix} & \text{if } S_{D1} < 0 \\ \begin{bmatrix} s_{\tilde{\Theta}} & c_{\tilde{\Theta}} \\ c_{\tilde{\Theta}} & -s_{\tilde{\Theta}} \end{bmatrix} & \text{else} \end{cases}$$

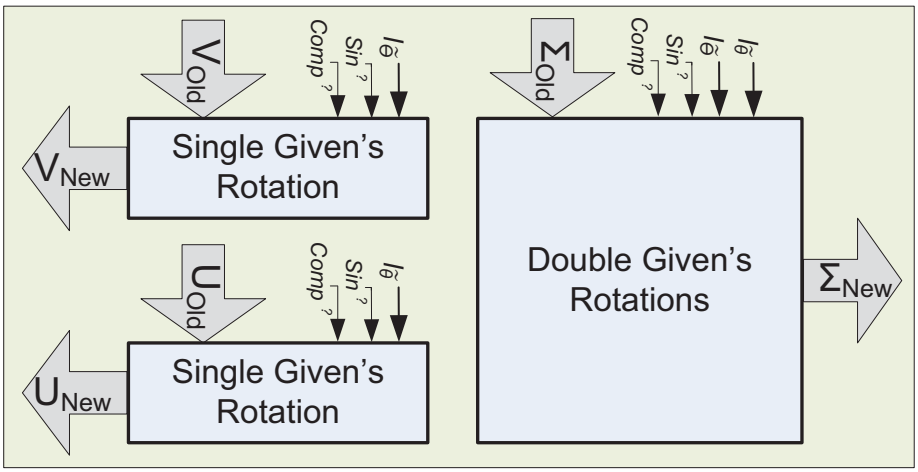


Figure 5.8: Diagram of the circuit for applying rotations

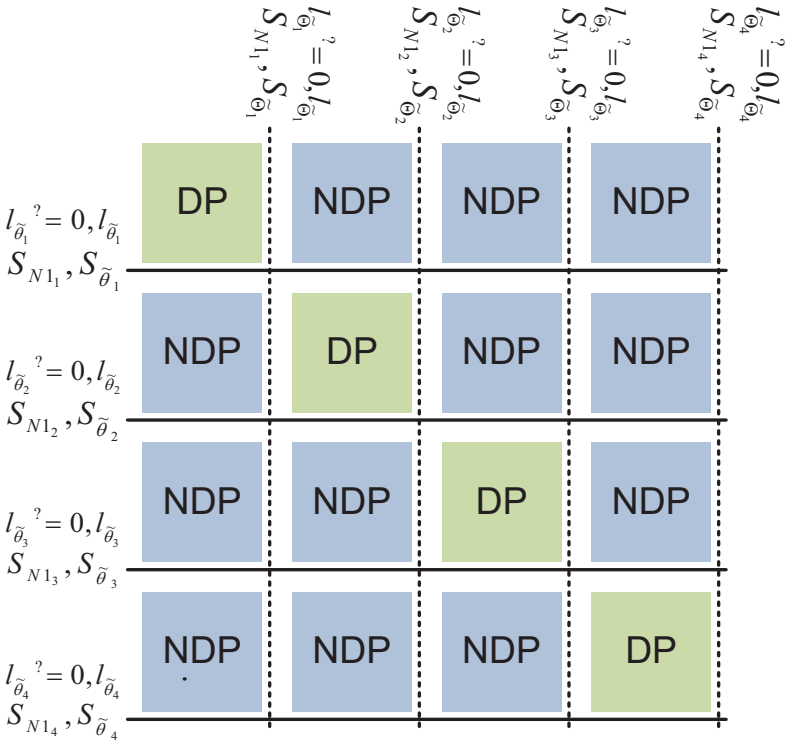


Figure 5.9: Architecture of the design.

5.4.2 Extending on the Base Design

The idea in this design is to add minimum required hardware to the SVD architecture to enable this design in performing the required task. To keep the design simple and also to be able to move the values between registers we decided to store the values in the registers with their red world fined point representation (keeping real world value). This means the SVD needs. to be implemented with 24Sfix18; while, we would be able to implement SVD processor with only 20 bits without having to keep the real world values (RWV) in the registers. This also means, all the registers need to be implemented with 32 bits of width. The maximum required precision is 21 bits and the maximum required integer bits are 10 bits, So the registers are implemented in 32Sfix21.

Matrix and Vector Multiply

Matrix and vector multiplications are the most frequent operation in this process and they need to be implemented very efficiently to make the system effective. Each vector needs to be multiplied by all the elements of the matrix vow vise and then the results are added together column vise. This process require 64 multiplier and eight adder tree with eight in pat in o aw design. Vector by matrix multiplication takes one clock cycle. The matrix multiplication is implemented using eight vector multiplication. In each vector multiplication a column of U^T (a row of U) is multiplied by the matrix S using 64 multipliers and a row of $U * S$ is generated using an adder three via each column of S. Dual to this process is when S is multiplied by a row of V (a column of V^T) and the adder three

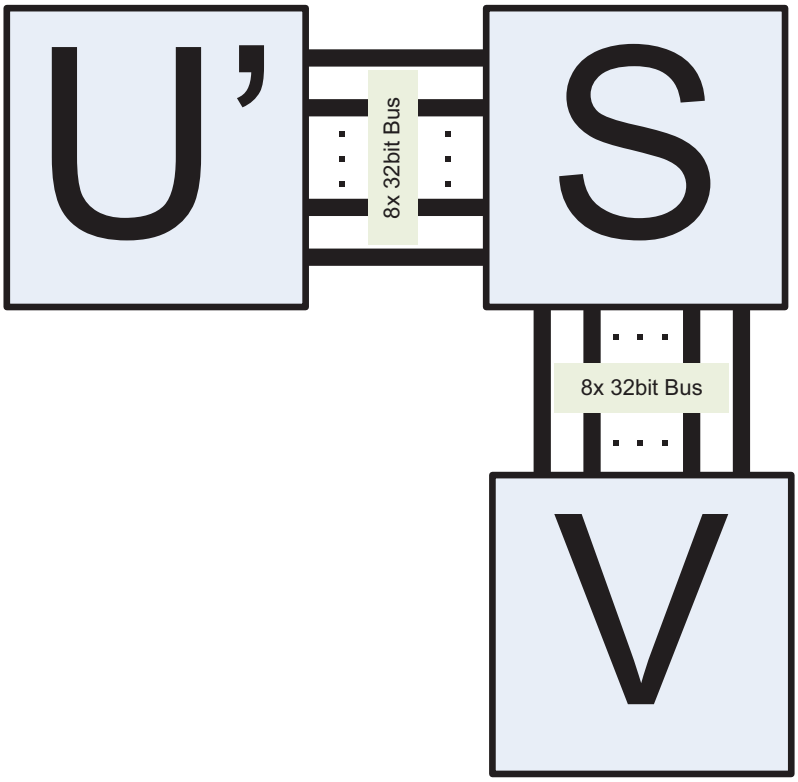


Figure 5.10: Structure of the bus.

through each row generates a column of $U * S$. This operation requires 64 multiplier, and 56 adders. This operation takes eight clock cycles. Figure 5.10 shows how vertical and horizontal bus extended to enable this operation.

Multiply and Add

The processor is capable of multiplying a vector by matrix in one operation. To add the multiplication results together we one using an adder tree of seven add us; however, if we make these adders re-arrangeable we can use them to perform multiply and accumulate/add. With adding an extra adder and reusing the components already excited we were able to provide multiply and add in one operation. This hardware is also used for implementing element wise multiplication (gain), and element wise addition. In each case the unused inputs are connected to constant zero and one respectively.

Special case Vector Multiply

The **Step 5** and **Step 6** in Algorithm 9 is treated as an special case. This process multiplies a vector by matrix and the result is multiplied by a vector to achieve a cost value. This requires two operation for each vector in candidate list. The complexity of this operation considering the size of the candidate list(64 vectors) is considerable however, the values of vector **t** are integer numbers in range [-8,8) which only requires four bits for representation. The idea is to design a hardware that is able to perform eight vector by matrix multiplication in one clock cycle in this special case. Accomplishing this idea require a small change in the bus architecture as well as the multipliers.

The bus design has to be able to distribute every 4 bits of the vector **t** to d different row of the horizontal and vertical bus system (every vector is stored in one register of size 32 bits). This provision will also help when sort system is sorting the cost value of and its associated vector (keeps the vectors and their required space to a minimum). Multipliers need to be able to perform a 32×26 (32Sfix21 multiplied by 26Sfix18) multiplication and only keep the values of bits in range 32Sfix21. Instead of implementing a 32×26 bits multiplier we implement each multiplier from eight, 4×26 bit multipliers and a re-arrangeable adder three. This will increase the hardware complexity but being able to calculate eight cost values in parallel is critical in efficient hardware implementation of this algorithm. This provision enables us to calculate the cost value for eight vector of candidate list. The adders that generate a 32 bit multiplier from eight 4 bit multipliers are reused in this case to make this possible.

Inverse Operation

This operation finds the inverse ($\frac{1}{x}$) of the values in register file S and stores the final results in S. To enable calculation of multiple iteration of this function for more accuracy we need to add two register vectors, $\bar{s}$, and $\bar{\bar{s}}$. Each register vector is made of eight register of size 32 bits. The operation changes the values of register vectors, $\bar{s}$, and $\bar{\bar{s}}$. Inverse uses the tailor series expansion to calculate $\frac{1}{x}$ using various number of multiplication. We use Algorithm 11 to calculate the various elements of one of two Taylor series expansion ($\frac{1}{x} = \frac{1/N1}{1-((N1-x)/N1)}$ or $\frac{1}{x} = \frac{1/N2}{1+((x-N2)/N2)}$), where $v_1 = ((N1-x)/N1) < 1$, $v_2 = ((x-$

$N2)/N2) < 1$, and $N1, N2$ are power of two integer numbers bigger and smaller than x respectively. The algorithm uses the smaller of the two $|v_1|$, and $|v_2|$ ($|v| = min(|v_1|, |v_2|)$), and the sign of v is the sign of the one that has smallest absolute value. In this case in the first clock cycle $\frac{1}{x}$ will be estimated two the first two element of $\sum_0^n v^n$ based on the algorithm and $\bar{S}$ and $\bar{\bar{S}}$ will hold the value of e and $e-n$; the following clock cycles will add one element of expansion to the equation (for second clock cycle $\frac{1}{x} \approx 1+v+v^2$); while, $\bar{S}$ will hold v and $\bar{\bar{S}}$ will hold s^n.

Algorithm 11 Algorithm to Calculate the Inverse.

Input: x, per.
Output: e, e_n, x^{-1}.
Step 1: Calculate the sign of the input:

$$S_x = Sign(x)$$

Step 2: Calculate the equivalent integer value of input value using following formula:

$$X = abs(x) \times 2^{per}$$

Step 3: Calculate the $exp(x)$ most significant bit using the following case statement:

$$(exp(X), v) = \begin{cases} (0,0) & \text{if } X = 0 \\ (\lfloor log_2{}^{|X|} \rfloor, 1) & \text{if } X \neq 0 \end{cases}$$

Step 4: Calculate the closest power of two number to X:

$$Exp(X) = exp_2(X) + \begin{cases} 1 & \text{if } 1.5 \times X > 2^{exp_2(X)+1} \\ 0 & \text{default} \end{cases}$$

Step 5: Calculate the error value:

$$Er = 2^{Exp(X)} - X$$

Step 6: Calculate the outputs:

$$e = Er \times 2^{-Exp(X)}$$
$$e_n = e \times 2^{per-Exp(X)}$$
$$x^{-1} = 2^{per-Exp(X)} + e_n$$

In this algorithm precision (per) is mentioned as an input while in fact this in hardware implementation this constant is known. The $Sign()$ function calculates the sign of the input and assign one for $input \geqslant 0$ and zero $input < 0$. The $|\quad|$ function calculates the absolute value of its input argument. The multiplication in the Step 2 of the algorithm should not

Table 5.3: Taylor series implementation aspects.

# of elements	# of multiplier	maximum error	RMS of error ratio
1	0	33.34%	0.1964
2	0	11.11%	0.051
3	1	3.7%	0.0145
4	2	1.23%	0.0043
5	3	0.41%	0.0013
6	4	0.14%	0.0004
7	5	0.046%	0.000123

be implemented and it just shows the designers should look at any input value as a integer number. The Step 3 in the algorithm is a priority encoder, and Step 4 is implemented using an adder, a comparator, and the priority encoder of the Step 3 is extended to MSB ($2^{exp(X)}$) and $2^{exp(X+1)}$. Table 5.3 demonstrate the implementation error for different number of elements in the Taylor series expansion. First column is the number of series elements used, the second is the number of multipliers used, the third column is maximum possible error, and the final column if the RMS of error divided by actual number in a 10000 randomly generated data in a normal distribution. In this table $per = 32$ to eliminate the quantification error effect.

Matrix Inversion

In [67] the process of calculating matrix inverse using SVD of a matrix and two matrix multiplication (pseudo inverse) is explained. As demonstrated in Figure 5.10 the architecture of b us allows for efficient implementation of this process. In this process we first calculate the SVD ($A = U \times S \times V^T$) of the input matrix, then using the method mentioned in previously we calculate the element wise inverse of diagonal matrix $\mathbf{S}$ (S^{-1} is the matrix

with element wise inverse in diagonal elements), and finally calculate $A^{-1} = V \times S^{-1} \times U^T$. The number of required inverse function iterations and also the precision requirements for both matrix inverse process are mentioned in Table 5.2. To calculate the pseudo Inverse of matrix $\mathbf{A_{IF}}$ ($\mathbf{A_{IF}}^{-1}$) seven SVD sweeps and 1 cycle (only first to element of Taylor series expansion) for inverse function is required. To calculate the pseudo Inverse of signal matrix $(\mathbf{H} * \mathbf{H}^T + \sigma/P\mathbf{I})^{-1}$ eight SVD sweeps and 7 cycles for inverse function is required.

Sort

The sort ability is added to the processor to enable the sort required in **Step 3**, and **Step 7** of the Algorithm 9. This function is implemented using nine sort net work each able to sort light input in a single a lock cycle. We use parallel bible Sort and these sort net work, in conjunction with sort the values in sort buffer. The sort buffer does benefit from a two layer design. The two layer design enables the buffer to sort the vector costs and at the same time re-position the associated vectors in the buffer when the **Step 7** of the algorithm. This two layer design also gives us the ability to keep the sorted vectors and their associated costs in the second layer while performing **Step 3** sort for iterations other than the first iteration. The precision required by sort in **Step 3** is 9Sfix4, and it is 9Sfix8 for **Step 7**. Sort buffer in the only memory element that holds signal with two different RWV in it. This provision helps effectively in reducing the size of sort networks. The size of the sort buffer is 72 registers and the full Sort takes 17 clock cycles. In **Step 3** of algorithm we only need to sort 64 elements; also, the full Sort is not required in **Step 7**.

GE

We are implementing the GE with row pivoting. Implementing the partial pivoting would result in reduced performance but it is still in the acceptable range. There are various method for practically implementing the GE. The simplest method is to use multiply and add to zero-out all the elements below the main diagonal. for example to zero *c* in equation (5.10, assuming None of the elements are zero, we multiply the first row by *c* and multiply the second row by *a* and subtract the results from each other. The result would replace the second row of the matrix. The only problem is the Large dynamic range of the values in this method. Another method that is able to keep the dynamic range limited is using dividers to normalize the numbers. In the same example we have to divide the first row by *a* and multiply it by *c* at the same time, then subtract the result from the second row and replace it with the original valve of the second row. The later method however, require the use of dividers or it will take multiple cycle normalizing the values before zeroing the values of the elements below the main diagonal in each column.

The third option, that we are utilizing, is to employ the first method and every time divide the result by the closest power of two bigger than *c*. Finding the divisor in this case require a simple circuit (similar to a priority encoder) and applying the division only needs a barrel shifter. The dynamic range in this case is 16 bits (16Sfix9). Utilizing this technique require two multiplier for every element in each row below the main diagonal. This specially in the first cycles of elimination process require more number of multipliers

than we have available. To solve this problem we need to re-arrange the 32 bits multipliers to two 16 bits multipliers.

5.5 Hardware Implementation and Results

We have implemented the hardware presented in previews section in TSMC 90 nm standard cell library using Synopsys design compiler. The results are presented in Table 5.4[6]. Our goal was to found a state of the at implementation of the two other category of detectors and compare the results. Finding a fair reference for companion is challenging in this case. The linear receivers have been the subject of research for more than 50 years. The most resent works are trying to implement such detectors, so that they meet the specific sequin meat of a certain standard, [70], and [71]. A comparison between multiple MMSE detector is presented in [72]. The search based detectors on the other hand tend to use the gain achieved by other orthogonal techniques to compensate the loss due to over simplification of search hardware. This includes the gain achieve by Soft output implementation or iterative detection and decoding to (soft input, soft output) compensate the performance loss due to hardware implementation [73], and [74].

In this work we consider the joint effect of detection and soft decoding in MIMO receivers. This gives a new boundary for what is called "near-ML" performance. A design that is able to provide near-ML performance (typically 0.1 dB or less) in an uncoded performance curve, might have a completely different curve after decoding. This decision is

[6]The value inside the parenthesis are scaled to 90 nm technology.

made to be fair regarding set output generation. The soft output generation in linear detector is straightforward and require less hardware complexity; however, the soft output calculation in search based detector require the calculation of MAP values which demand more hardware. In addition, the MAP values generated should have enough accuracy to prevent performers loss. The MAP performance curve demonstrated in Figure 5.2 is achieved from a detector with depth-first tree search sphere decoding algorithm and list size of 32. The LLR values in this are saturated in [-8,+8) range. The LDPC with rate of 1/2 is used for decoding and in this case our Monte Carlo simulations do not have any performance gap with the best achievable results. The ML curve in this case shows the achievable result for hard output detection.

The last point to be considered in regards to fair comparison of hardware implementation is the flexibility of hardware. Even though we optimized and study the performance and designs for 4×4 at over arrangement and 16 QAM, our design is able to receive any antenna arrangement up to 4×4 and any QAM modulation up to 256 QAM. Eberli et al. in [75] present an adaptive MMSE detector that is able to process 4×4 antenna arrangement; however, it fails to meet the standard latency requirement. Table 5.4 shows the hardware implementation of an MMSE detector. The design is implemented in 40 nm CMOS technology and provides up to 468 Mbps (208 if scaled to 90 nm technology) throughput, and has 550 KGE (GE is the hardware complexity of one two input Nand gate) complexity. The normalized throughput in this case is 6.98 times our IF design and 3.1 times if we scale the results to 90 nm technology.

In [76] hardware complexity of an ML-APP solution for 4×4 16 QAM arrangement is reported. This design uses 180 nm CMOS technology provides a throughput of 38.4 (76.8 if scaled to 90 nm CMOS technology), while it requires 32.7 W power. This will require 111.73 times more energy per bit (when this design is scaled to 90 nm technology) compared to our IF detector. A first order estimation of a "near-ML-APP" solution has been provided in this reference that uses spherical decoding algorithm and is able to reduce the hardware complexity to 3.7 percent of the original design. Two near-ML hardware architectures been presented in [77], and [70]. While both designs use K-best algorithm, the work in [70] is the most resent design and also better choice concerning a fair comparison since it provided a flexible kernel capable of detecting multiple antenna and modulation arrangement. The throughput is calculated for 16 QAM and 4×4 antenna arrangement based on the proposed architecture. This design require 54.3 times more energy per bit and has normalized throughput of 12.8 percent compared to our IF design. This results are very promising for IF detector since it is able to reduce the gap between MMSE soft output detector and ML-APP detector to 48% of the original gap and only demand 3 times more complexity than MMSE detector. While a near-ML detector demands 24.2 times more complexity compared to MMSE detector to reduce the gap to 7.8% (2 dB performance loss in our study) of the original gap.

Table 5.4: Different detectors hardware specs.

	This Work	[71]	[77]	[76]	[76]	[70]
Detector Type	IF	MMSE	K-best	Exhaustive Search	Spherical	K-best
Technology (nm)	90	40	180	180	180	90
Clock Frequency (MHz)	95.2	320	47	122.88	122.88	166.67
Throughput (Mbps)	109.6	468 (208)	18.8 (37.6)	38.4 (76.8)	38.4 38.4 (76.8)	8.8
Power (mW)	128.9	-	-	32700 (10092.6)	-	562
Area (mm)	5.38	-	-	269.36 (67.34)	10 (2.5)	7.29
Gate Count (KGE)	896.6	550	300	-	-	2062
Energy per bit (nJ/bit)	1.176	-	-	851.56 (131.4)	-	63.86
Normalized Throughput (Mbps/KGE)	0.122	0.851 (0.378)	0.62 (0.125)	-	-	0.0156

5.6 Conclusion

In this work we present the hardware implementation of Integer Forcing detector. This design uses singular value decomposition (SVD) for calculation of Eigen values, Eigen Vectors, and matrix pseudo inverse. Since SVD is used as the basic building block for all these functions we developed a separate processor to calculate SVD in parallel with other tasks. The proposed architecture extends the capabilities of SVD hardware evolves it to an Integer Forcing (IF) decoder detector. In the next step we separate this special propose processor to three processors, SVD, Channel, and symbol processors and explained the requirement of each processor. The channel processor does handle all the calculation required to achieve a full rank receiver matrix except the SVD part. The SVD processor calculates the Eigen vectors, and is used in the process of calculating the pseudo inverse of two 8×8 matrices. The symbol processor process each symbol (144 per packet) received from the channel using the receiver matrix to recover the transmitted data. This detector is implemented in 90 nm CMOS technology. We compared the performance and hardware implementation cost of this detector with two state of the art detectors (a linear and a close to maximum likelihood detector). Our result has 13.5 dB better performance compared

to the state of the art MMSE detector while it requires only 3 times more hardware complexity, and demands 8 times less hardware complexity than a state of the art near-ML detector. The near-ML detector provides 11 dB better performance than our design and 24.5 dB better performance than an MMSE detector. This design is based on a special purpose processor design and does not employ pipelining which provides the hope for better hardware efficiency for our future work.

6 ASYNCHRONOUS BASE BAND PROCESSOR FOR COOPERATIVE MIMO IN SATELLITE COMMUNICATION*

The challenges in satellite communication (SatCom) include but not limited to the customary complications of telecommunication such as channel condition and signal to noise ratio (SNR). SatCom system is also prone transient and permanent radiations hazards. Hence, in spite of the harsh environmental factors, (weather phenomena, solar events, etc) a SatCom system must maintain reliable and predictable communication functions with limited source of power. This work presents a SatCom system design for achieving both low-power and high fidelity communication. The design uses cooperative multiple input multiple output (MIMO) for spectral efficiency and diversity, low-density parity-check (LDPC) decoding for near Shannon-limit gain, and dynamic voltage and frequency scaling (DVFS)-assisted asynchronous circuit designs to achieve low-power and fault tolerance. The MIMO system permits uninterrupted service in the event of temporary/permanent link or unit failures. The results shows the perfect tolerability against the radiation that apply upto 25 fQ on critical path which is more than 600X the minimum charge necessary to flip a gate output.

*Reprinted with permission from "Asynchronous baseband processor design for cooperative MIMO satellite communication" by Ehsan Rohani, JingWei Xu, Tiben Che, Mehnaz Rahman, Gwan Choi, Mi Lu, 2014, IEEE 57th International Midwest Symposium on Circuits and Systems (MWSCAS),833-836, Copyright 2014 by IEEE.

6.1 Introduction

MIMO has been widely acclaimed due to its high throughput and numerous advantages, such as spatial gain, diversity, and interference reduction with no additional cost from the perspective of transit power and bandwidth [78], [3]. It has already been adopted by many terrestrial standards, such as the IEEE 802.11n, 802.16e, Long Term Evaluation (LTE) etc. In effort to keep the pace with terrestrial systems, SatCom systems are also trying to incorporate MIMO for higher data rate and bandwidth efficiency.

The general setup of MIMO Satellite uplinks and downlinks is proposed in [79], whereas the optimization for maximum achievable channel capacity is done based on Line-of-Sight (LoS) signal component, which is the backbone of SatCom. A number of MIMO SatCom application examples, including the common case of satellites with transparent communication payloads are discussed in [80]. It showed that the construction of MIMO SatCom with optimal capacity is practically possible under the assumption of undistributed LoS propagation. However, with severe weather condition (such as rain and wet snow) the MIMO Satellite channel capacity degrades significantly [81]. To solve this issue, several techniques such as dual polarization, power allocation with linear precoding etc. have been proposed [82]. Cooperative satellite communication is possibly a better solution to this problem [83], offering extended satellite coverage in uncovered areas. Generally, cooperative systems have a source node multicasting a message from a source node to a number of cooperative nodes, which in turn resend a processed version to the intended destination

node. Along with this classical concept of repetition based forwarding, several techniques have been discussed in [84], [85] for cooperative MIMO SatCom. However, most of these cooperative proposals require symbol-level synchronization between cooperative nodes. The lack of synchronization may result in inter-symbol interference and dispersive channels. To address this issue, asynchronous cooperative MIMO communication techniques is proposed [84].

From the perspective of transmission energy consumption and computation, the use of MIMO can save a significant amount of transmission power, or can increase the bandwidth even considering the local energy cost for trafficking joint information within the cooperating nodes [79--84]. In [85] the power efficiency of the cooperative MIMO in term of transmission robustness has been studied. The result shows that,if the target distance is more than a threshold or channel condition is poor, use of MIMO is inevitable. Although MIMO system with increased circuit complexity can achieve better performance in term of power due to its diversity and power gain, decreasing the energy consumption of the decoding circuit itself can lead to another venue of exploration. However, most of the MIMO decoding circuits are sensitive to clock, i.e., synchronous. As a result, they are very sensitive to antenna placement and radiations effects, leading to low reliability, less tolerability and very high bit error rate (BER).

With the emphasis on low power, we developed a fault-tolerant MIMO receiver for satellite transmission using the approach of asynchronous circuit design. The key attribute of an asynchronous circuit is remarkably high reliability even at very low power consump-

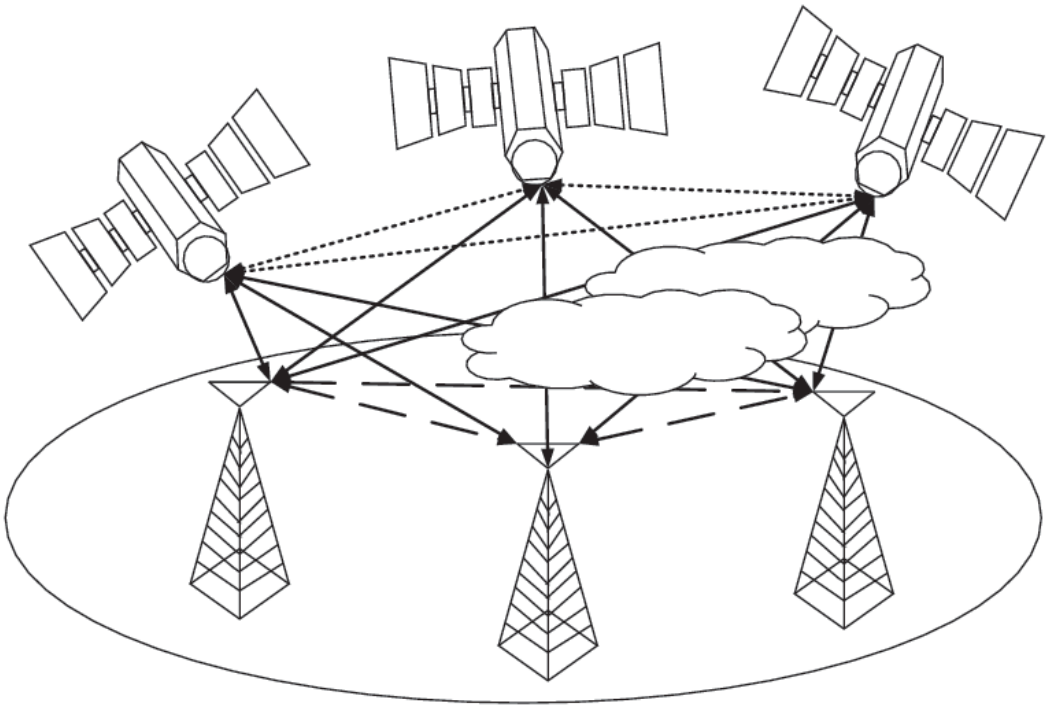


Figure 6.1: System overview of cooperative MIMO satellite cfommunication

tion. These circuits naturally perform computation on-demand until correct completion. Moreover, in terms of stability, asynchronous circuits are found to be inherently more robust [86]. Later, research on the implementation techniques of asynchronous circuit proposed 100% fault tolerant circuit for Wireless MIMO communication. Several ultra-low power design techniques have been reported in [87, 88].

The overview of the cooperative communication scenario is presented in Fig 6.1 which includes, e.g., 3 cooperative satellites along with 3 antennas on earth. Due to the cloud representing severe bad weather condition, the MIMO Satellite channel capacity is degraded providing low tolerability and high BER. The proposed idea of combining cooperative MIMO SatCom with asynchronous decoding circuit is possibly the ideal solution to this

circumstances. Cooperative MIMO SatCom can ensure coverage even at the failure of one satellite. Moreover, use of asynchronous circuit [88] can guarantee 100% fault coverage with ultra-low power consumption and higher reliability even at very low voltage levels.

We have synthesized the MIMO receiver and conducted *SPICE* simulation on critical path to show that the perfect performance is achievable even in the worst case scenario, with fault injection charge level that is 625 times greater than minimum charge required for flipping a gate in CMOS circuits. Moreover, the effect of reducing the supply voltage in both present and absence of radiation is studied. This shows that with correct power management method, the effect of radiation can be removed while still keeping the power consumption at minimum. The cooperative asynchronous MIMO design can save approximately upto 90% power for baseband processing.

The rest of this work organized as follow: The overview of the system is given in the Section 6.2. In the following, the modeling and setup of simulation are presented in Section 6.3. After analyzing the results from simulations in Section 6.3, this work is concluded in Section 6.4.

6.2 Overview of System

Here, we present an asynchronous MIMO processor for cooperative satellite comunication with detector as a LORD detector [89]. We have used a fully parallel LDPC decoder in this study while autos believe same result can be expected from more advanced

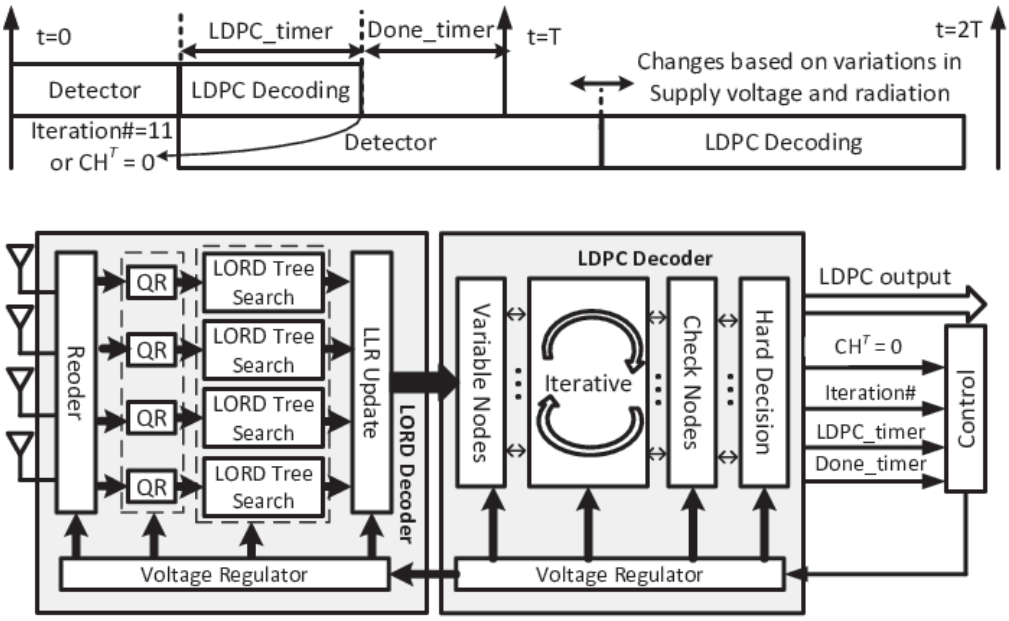


Figure 6.2: Timing and block diagram of asynchronous MIMO receiver

architecture [90]. The architectures of both designs are specifically optimized for minimum performance loss. The receiver not only uses the essential characteristic of MIMO to increase spectral power efficiency, but also, uses the properties of asynchronous circuit to overcome the faults introduced to the circuit by radiation. DVFS method is applied to increase the system flexibility in different environment while keeping the power consumption at minimum level. In order to meet the real time requirement of communication system, a deadline is applied to the calculation time of each frame which we will show its effect is negligible due to the error tolerance of the receiver and the iterative properties of decoder. The target throughput of the system is 200 Mbps. Fig. 6.2 shows the timing and block diagram of receiver. Both LDPC and detector are presented using the logic of asynchronous circuit. The detector starts right after a *Hard deadline* and as soon as the detector

calculates the first frame, the decoder can start decoding. The process ends when LDPC reaches the maximum allowed number of iterations, or all the check nodes are satisfied. The other factor that can terminate the LDPC routine, is reaching the *Hard deadline*. In case that a *Hard deadline* (upward arrows in timing diagram of Fig. 6.2) is reached, the LDPC stops and delivers the last updated values output.

In block diagram of Fig. 6.2, the overall architecture of the system is presented. As depicted, LORD detector is the first step in base band processing and it is connected to LDPC decoder. The buffers that are the interfaces of the detector and decoder, are not presented in the figure. Decoder uses layered iterative decoding method and after each iteration it checks to see if all the check nodes are satisfied. Right before the decoding process ends, a hard decision on the LLR values will be applied. The DVFS control unit uses some characteristic of LDPC decoder to calculate if the voltage should be increased or decreased. In Section 6.2.3 we will describe the function of DVFS control unit in more details.

The ratios in the picture are not relative to what happens in reality and it is just for demonstration purposes. With this design there is a theoretical possibility in that the third detector process finishes before first LDPC decoding process is finished. This requires increase in buffer size and that directly translates to power consumption. We noticed that in practical changes in the system delay will not change dramatically and this situation will not happen. A preventive measure is also taken so that the third detection process does not starts unless the first decoding process is finished to address the theoretical concern of

the problem. As, this will not affect the cases that we are simulating, the results is not presented in interest of space. A fair notice about the figure is that, the detector needs to calculate multiple symbols ,one frame, before LDPC can start working on that frame. Synchronous to asynchronous (STA)and asynchronous to synchronous (ATS) buffers are necessary before the reordering unit and after hard derision unit respectively. The clock of ATS buffer is equal to hard deadline.

6.2.1 MIMO System

The transmission model of a MIMO system with M transmit antenna and N receiving antenna can be represented as $y = Hs + n$, where s is the symbol of $N \times 1$ dimensional transmitted signal, H is $M \times N$ complex channel matrix, y is symbol of M dimensional received vector, and n is a $M \times 1$ vector of additive complex symmetric Gaussian Noise. The entries of s are chosen from a set of complex constellation (Ω). This work works with 4×4 MIMO arrangement with receiver having the knowledge of channel matrix and variance of noise. For MIMO detection, one approach is to search exhaustively among all the constellation points, known as maximum likelihood (ML) by calculating: $\hat{s} = min \parallel y - Hs \parallel^2$, where $\parallel \cdot \parallel$ denotes the 2-norm. However, sphere decoding can successfully reduce the search space by evaluating only those points, which fit inside a sphere around the received signal and it can be formulated as:$\hat{s} = argmin \parallel y - Hs \parallel^2 : d(s) < r^2$. This problem of decoding can also be formulated as tree search problem, where each branch is one of the possible transmitted symbols.

6.2.2 Asynchronous Design

The receiver with pre-charged static logic (PCSL) is presented in [91]. Here transistor is added to each static gate to enable the pre-charge sequence and the gates of those transistors are connected to request signal (Req.). In the evaluation period each gate works as a static logic gate, which has two inverters that specify if the result of the gate is correct or not. At each stage of the design, Req. signal is received from previous stage, and the acknowledgement signal (Ack) is sent back after the processing ends. In this design, instead of using the concept sub-threshold voltage, supply voltage is varied from 0.7 V to 1.1 V and, 0.83 V is the maximum source voltage for system in absence of radiation. The maximum charge, expected to be 25 fQ, is applied to the specific nodes of critical path is 625 times higher than the minimum charge (40 aQ) required to flip a bit in 45 nm technology.

6.2.3 DVFS Control Unit

The DVFS control unit uses a simple algorithm to determine if the voltage should be increased or decreased. The system, as shown in Fig. 6.2, uses continuous on-chip voltage regulators based on the command received from the control unit and their voltage levels change from 0.4 V to 1.1 V. This amount of change in supply voltage can theoretically provide up to about 4 times the speed scaled for the circuit. The control unit uses four signals to determine whether to increase or decrease the voltage. These four signals are, $CH^T = 0$, Iteration#, LDPC_timer, and Done_timer. $CH^T = 0$ is the signal that specifies if all the check nodes are satisfied in the last iteration of decoding. C is the matrix of codes

and H is a parity check matrix. If $CH^T = 0$, it means that all the outputs are valid codes (does not ensure the correctness for transmuted data). Iteration# is an eight bit bus which specifies the number of LDPC iterations. LDPC_timer calculates the time from the start of LDPC decoding until the end of it. Done_timer indicates the time from termination of LDPC iterations until next *Hard deadline*. Control unit starts with checking if $CH^T = 0$ is satisfied; if so, it means that all the necessary calculations have been done in time. In this case, control unit only has to check if there is a room for reducing supply voltage. If $CH^T \neq 0$, the control unit checks Iteration# and a request for increase in voltage is made unless the Iteration# is less than 11 (maximum allowed iteration). If the number of iterations is already 11, the control unit checks for the possibility of voltage decrease. This basically means if the available time is bigger than one LDPC iteration compared to the last received data, operating in the same scenario.

6.2.4 LDPC

An LDPC code is defined with matrix called H. Each row of the matrix is a parity check equation and columns are associated with received bits. Using Tanner graph the parity check equation can be called check nodes and the coded bits can be presented by variable nodes. A variable node is connected to a check node if the associated bit in H matrix is one. The process of decoding can be done by passing information iteratively through the edges of the graph. The LDPC used for this experiment is a fully parallel soft LDPC decoder. This LDPC uses H matrix of 2304 with 1/2 rate coding scheme. Although receiver applying

more than 11 iteration is not necessary, with increase of supply voltage it can accommodate 32 iterations in radiation free environment and uses 38.3 mW. The throughput is 200 Mbps. This LDPC has two timers which are the only synchronous circuits of the design. One timer is used to keep the time from the start of LDPC iterations until the end. The other keeps the value of timer from the end of iterations until next *Hard deadline*. Occasionally, second set of timers may be needed since the second frame may finish detection much earlier than the first frame.

6.2.5 LORD Detector and LLR Update Unit

The LORD algorithm is one of the best available MIMO detection methods in term of performance to consumed power ratio [89]. This detector provides soft output while at the same time keeps the calculation complexity minimum. The throughput and completion delay of this decoder unlike many other detectors (depth first search algorithm) is constant. The detector in this study is designed for 4×4 MIMO system with 16 QAM modulation. The detector searches all branch of the tree at first layer and takes the best child of each branch for rest of the layers. To provide a reasonable list of candidate that is able to estimate the log likelihood ratio (LLR) values for soft decoding, LORD has to reorder the channel matrix for all levels of the tree and finds the best candidate each time. The detector uses 6.8 mW while providing 230 Mbps. The proposed design is to accommodate 15 percent increase in delay caused by the radiations of 0.9 V supply voltage. The output of LORD tree search unit will be delivered to LLR calculation unit which calculates the soft values

for decoder.

6.3 Setup and Simulations

The radiation that satellites are in contact with can hit the circuit of receiver and introduces charge. If the charge is high, it will cause permanent damages which are out of the scope of this work. The charges not forcing permanent damages, may cause fault in the synchronous circuit. Certain asynchronous circuits are designed to eliminate the faults by 100% in cost of increasing delay. The design of receiver is able to tolerate the charges that cause around 60% delay, while the supply voltage is 1.1 V. The study of the effect of charges applied to the critical path is presented in section 6.3.1.

For a specific throughput, the time to process one MIMO symbol is pre-determined and depends on delay caused by radiations and supply voltage. As a result, the number of LDPC iterations will be affected by the increase of computation delay for both MIMO detector and LDPC decoder. The decoding fidelity over SNR with different radiation and supply voltage is simulated using Matlab, as presented in section 6.3.2.

6.3.1 Study of Critical Path

The design of the system allows to tolerate any change in delay as long as it is less than 50% without degradation in performance, while the power supply value is 1.1 V. To set up the exploration on relationship between error injection and delay, the RTL coding for the receiver is synthesized using *Synopsys Design Compiler*. The critical path is extracted in

SPICE netlist by *Synopsys Primetime*. The error model presented in [92] including dual exponent current injection model, is used in our experiments, where the radiation is modeled as a current source connected to nets. For a certain level of radiation, a quantity of electric charge at random rate will be injected to a random net during the transition time. Two constraints are applied to the delay simulator. First is to pick up the fault injection time points, so that the resulting delay can be propagated to the output of the circuit. Secondly, the whole process needs to be done in such a way that delay never decreases, thereby simulating the worst case scenario. In order to obtain the effect of radiation on delay, SPICE simulations are run on extracted critical path for both detector and decoder to examine the delays at different levels of electric charges. Fig. 6.3 shows the average delay caused by different charges in described simulation settings. The results are presented in percentage. The effect of the charge on each circuit is different because of their differences in architecture and sizes of the gates. The maximum charge applied to the circuit causes around 12% delay and it is more than 600 times the minimum required to flip a gate value in 45 nm technologies. The system will lose performance only if the supply voltage is less than necessary value. Next we will study the results of power and performance of the system for different voltages and radiation effects.

6.3.2 Results

This system has the ability to cope with very harsh environments as well as can reduce the power consumed effectively. In very high SNRs and low radiation environment, the

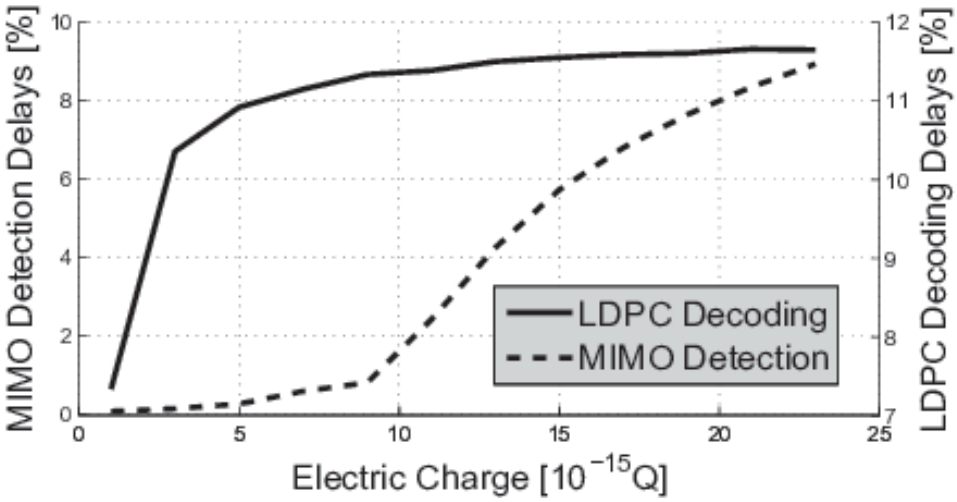


Figure 6.3: Radiation effects on critical path delay.

power usage of the system can be reduced to 3.2 mW with 0.7 V as power supply and very slow system. In this case, LDPC uses only one iteration to decode the received data and most of the available calculation time is dedicated to detector. To achieve the exact power numbers, the designs are synthesized using *TSMC 45 nm CMOS Technology* and *Synopsys Design Compiler*. Moreover, Matlab simulation is performed to calculate the BER of the system. These simulations are for 100000 frames (around 230 Mb) or 100 errors whichever comes first. The results of these simulations are presented in Fig 6.4. The left axis is presenting the BER while the right axis shows the power in mW. The power curves are doted lines for different voltages of power supply and different charges (presenting different radiation situation). The BER curves that are presented with solid lines have the same marker as their paired power curves. Presented curves are chosen to demonstrate the ability of system to reduce power consumption and keeping the performance perfect at the same time. The figure shows that even with power supply set to 0.9 V system can tolerate

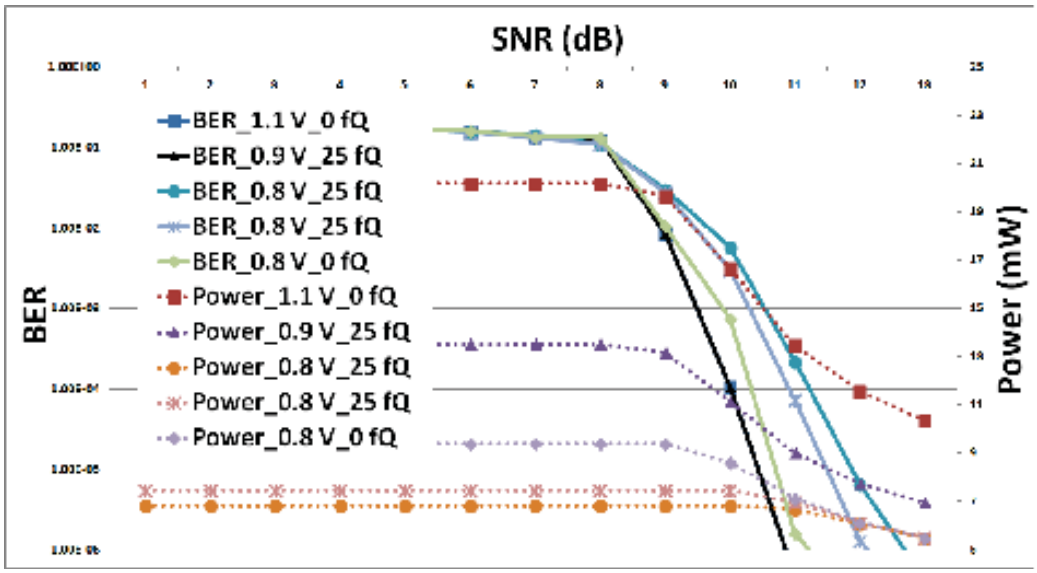


Figure 6.4: Power consumption and BER performance in different radiation and voltage scaling for SNRs range of 1 to 13 dB.

maximum delay caused by radiations. Essentially this can be inferred comparing the 0.9 V and 25f Q performance curve with that of 1.1 V and 0f Q as both have the same shape. The power consumption for 1.1 V curve is higher, but the time took by LDPC is less since in both situations, LDPC would have enough time to accommodate (which is necessary) the maximum of 11 iterations. The Fig. 6.4 also shows the effect of different charges on the performance of the system, when the supply voltage is 0.8 V. While the performance of the system in 0.8 V almost (0.1 dB difference) matches the maximum expected performance, it can cause more than 2 and 3 dB loss in performance for 10 and 25 fQ charge respectively. This shows that the mismanagement of supplying voltage can result in serious performance loss.

6.4 Conclusion

The employment of SatCom system includes overcoming the communication challenges such as bad channel condition, SNR etc. as well as maintaining reliable and predictable communication despite of bad weather condition even with limited source of power. This work refers to a co-operative MIMO with low power, fault tolerant asynchronous circuit design ensuring uninterrupted services even at the failure of one unit. The proposed idea of MIMO can exploit both spectral efficiency and diversity for achieving near Shannon limit gain, and use of DVFS-assisted asynchronous circuit includes the guarantee of low power consumption and 100% fault tolerability. The simulation results show the perfect tolerability as opposed to the radiation which applies up to 25f Q on critical path. This offers more than 600 times (40a F) the minimum charge necessary for flipping an output. Also results shows that the mismanagement of power supply can cause more than 3 dB performance loss.

7 SUMMARY AND FUTURE WORKS

This section includes a summary of our works in this dissertation, and the perspective of our future works.

7.1 Summary

Multiple techniques that are critical to advancement of telecommunication systems studies in this work. These techniques are iterative detection and decoding of the MIMO transceivers, its reconfiguration, Integer Forcing (IF) detectors, and the required hardware architecture. The investigation of the efficient hardware implementation of IF detector did result in an advancement in hardware implementation of singular value decomposition (SVD). This investigation started due to frequent use of SVD in IF detector, and we realized that efficient implementation of this detector is the key to improve the performance of IF detectors. In addition, we studied the asynchronous implementation of an iterative MIMO detector. The implementation shows a promising benefit in asynchronous implementation of these receivers for satellite communication (SatCom). The next section will summarize our future research plan.

7.2 Future Works

To advance the work presented in this dissertation a study on pipeline implementation and hardware reuse of SVD. We expect up to three time higher throughput in hardware implementation with minimum increase in combinational hardware. The hardware efficiency also would increase with the same rate. The SVD processor presented would be a suitable candidate for a fast MIMO-OFDM detector, and this needs to be investigated. The slowest-descent method used for determining the IF inverse matrix is currently occupying the major processing time and power of the detector; we expect an standard with larger frame size be able to better highlight the benefit of IF detector. In the IF detector architecture, the sort buffer is a hardware demanding part of the design and up to four parallel processor can share this part without any compromise. Four parallel processor that share this part would have higher hardware efficiency. The IF detector uses multiple multipliers while the SVD processor uses a series of adders to avoid the use of multiplier; a design that is able to merge these two hardware should be able to provide reasonable hardware performance increase. These mentioned tasks form our major plan to tackle this problems in near future.

REFERENCES

[1] ``Ieee standard for air interface for broadband wireless access systems," *IEEE Std 802.16-2012 (Revision of IEEE Std 802.16-2009)*, pp. 1--2542, Aug 2012.

[2] H. Kim, D. U. Lee, and J. D. Villasenor, ``Design tradeoffs and hardware architecture for real-time iterative mimo detection using sphere decoding and ldpc coding," *IEEE Journal on Selected Areas in Communications*, vol. 26, pp. 1003--1014, August 2008.

[3] E. Rohani, J. W. Xu, G. Choi, and M. Lu, ``Low-power on-the-fly reconfigurable iterative mimo detection and ldpc decoding design," *Applied Mechanics and Materials*, vol. 496, pp. 1825--1829, 2014.

[4] L. Wei and W. Chen, ``Integer-forcing linear receiver design with slowest descent method," *Wireless Communications, IEEE Transactions on*, vol. 12, pp. 2788--2796, June 2013.

[5] R. P. Brent, F. T. Luk, and C. Van Loan, ``Computation of the singular value decomposition using mesh-connected processors," tech. rep., Cornell University, 1982.

[6] J. Götze, S. Paul, and M. Sauer, ``An efficient jacobi-like algorithm for parallel eigenvalue computation," *Computers, IEEE Transactions on*, vol. 42, no. 9, pp. 1058--1065, 1993.

[7] J. Götze, ``Parallel methods for iterative matrix decompositions," in *Circuits and Systems, 1991., IEEE International Sympoisum on*, pp. 232--235, IEEE, 1991.

[8] E. Rohani, J. Xu, T. Che, M. Rahman, G. Choi, and M. Lu, ``Asynchronous baseband processor design for cooperative mimo satellite communication," in *2014 IEEE 57th international midwest symposium on circuits and systems (MWSCAS)*, pp. 833--836, IEEE, 2014.

[9] ``Ieee standard for information technology-- local and metropolitan area networks--specific requirements-- part 11: Wireless lan medium access control (mac)and physical layer (phy) specifications amendment 5: Enhancements for higher throughput," *IEEE Std 802.11n-2009 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, and IEEE Std 802.11w-2009)*, pp. 1--565, Oct 2009.

[10] I. W. Lai, G. Ascheid, H. Meyr, and T. D. Chiueh, ``Low-complexity channel-adaptive mimo detection with just-acceptable error rate," in *Vehicular Technology Conference, 2009. VTC Spring 2009. IEEE 69th*, pp. 1--5, April 2009.

[11] A. Burg, N. Felber, and W. Fichtner, ``A 50 mbps 4 times;4 maximum likelihood decoder for multiple-input multiple-output systems with qpsk modulation," in *Electronics, Circuits and Systems, 2003. ICECS 2003. Proceedings of the 2003 10th IEEE International Conference on*, vol. 1, pp. 332--335 Vol.1, Dec 2003.

[12] D. Garrett, L. Davis, S. ten Brink, and B. Hochwald, ``App processing for high per-

formance mimo systems [receiver symbol detector]," in *Custom Integrated Circuits Conference, 2003. Proceedings of the IEEE 2003*, pp. 271--274, Sept 2003.

[13] B. M. Hochwald and S. ten Brink, ``Achieving near-capacity on a multiple-antenna channel," *IEEE Transactions on Communications*, vol. 51, pp. 389--399, March 2003.

[14] J. Ketonen, M. Juntti, and J. R. Cavallaro, ``Performance-complexity comparison of receivers for a lte mimo-ofdm system," *IEEE Transactions on Signal Processing*, vol. 58, pp. 3360--3372, June 2010.

[15] C. Roth, P. Meinerzhagen, C. Studer, and A. Burg, ``A 15.8 pj/bit/iter quasi-cyclic ldpc decoder for ieee 802.11n in 90 nm cmos," in *Solid State Circuits Conference (A-SSCC), 2010 IEEE Asian*, pp. 1--4, Nov 2010.

[16] K. K. Gunnam, *Area and Energy Efficient VLSI Architectures for Low-Density Parity-Check Decoders Using an On-The-Fly Computation*. PhD thesis, Texas A&M University, 2006.

[17] K. Gunnam, G. Choi, W. Wang, and M. Yeary, ``Multi-rate layered decoder architecture for block ldpc codes of the ieee 802.11n wireless standard," in *2007 IEEE International Symposium on Circuits and Systems*, pp. 1645--1648, May 2007.

[18] I. W. Lai, G. Ascheid, H. Meyr, and T. D. Chiueh, ``Efficient channel-adaptive mimo detection using just-acceptable error rate," *IEEE Transactions on Wireless Communications*, vol. 10, pp. 73--83, January 2011.

[19] J. Boutros, O. Pothier, and G. Zemor, ``Generalized low density (tanner) codes," in *Communications, 1999. ICC '99. 1999 IEEE International Conference on*, vol. 1, pp. 441--445 vol.1, 1999.

[20] C. Studer, S. Fateh, and D. Seethaler, ``Asic implementation of soft-input soft-output mimo detection using mmse parallel interference cancellation," *IEEE Journal of Solid-State Circuits*, vol. 46, pp. 1754--1765, July 2011.

[21] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bolcskei, ``Vlsi implementation of mimo detection using the sphere decoding algorithm," *IEEE Journal of Solid-State Circuits*, vol. 40, pp. 1566--1577, July 2005.

[22] M. Myllyla, J. Antikainen, M. Juntti, and J. R. Cavallaro, ``The effect of llr clipping to the complexity of list sphere detector algorithms," in *2007 Conference Record of the Forty-First Asilomar Conference on Signals, Systems and Computers*, pp. 1559--1563, Nov 2007.

[23] A. Burg, M. Wenk, M. Zellweger, M. Wegmueller, N. Felber, and W. Fichtner, ``Vlsi implementation of the sphere decoding algorithm," in *Solid-State Circuits Conference, 2004. ESSCIRC 2004. Proceeding of the 30th European*, pp. 303--306, Sept 2004.

[24] M. Siti and M. P. Fitz, ``A novel soft-output layered orthogonal lattice detector for multiple antenna communications," in *2006 IEEE International Conference on Communications*, vol. 4, pp. 1686--1691, June 2006.

[25] P. Bhagawat, R. Dash, and G. Choi, ``Dynamically reconfigurable soft output mimo detector," in *Computer Design, 2008. ICCD 2008. IEEE International Conference on*, pp. 68--73, Oct 2008.

[26] J. Andrews, S. Buzzi, W. Choi, S. Hanly, A. Lozano, A. Soong, and J. Zhang, ``What will 5g be?," *Selected Areas in Communications, IEEE Journal on*, vol. 32, pp. 1065--1082, June 2014.

[27] ``Ieee standard for air interface for broadband wireless access systems," *IEEE Std 802.16-2012 (Revision of IEEE Std 802.16-2009)*, pp. 1--2542, Aug 2012.

[28] ``Ieee standard for local and metropolitan area networks part 16: Air interface for fixed broadband wireless access systems," *IEEE Std 802.16-2001*, pp. 1--322, 2002.

[29] A. Jovicic and P. Viswanath, ``Cognitive radio: An information-theoretic perspective," *Information Theory, IEEE Transactions on*, vol. 55, pp. 3945--3958, Sept 2009.

[30] Z. Yan, Z. Ma, H. Cao, G. Li, and W. Wang, ``Spectrum sensing, access and coexistence testbed for cognitive radio using usrp," in *Circuits and Systems for Communications, 2008. ICCSC 2008. 4th IEEE International Conference on*, pp. 270--274, May 2008.

[31] H. Arslan, *Cognitive radio, software defined radio, and adaptive wireless systems*, vol. 10. Springer, 2007.

[32] B. Nazer and M. Gastpar, ``Compute-and-forward: Harnessing interference through structured codes," *Information Theory, IEEE Transactions on*, vol. 57, no. 10,

pp. 6463--6486, 2011.

[33] K. Wesolowski, *Introduction to digital communication systems*. John Wiley & Sons, 2009.

[34] Y. S. Cho, J. Kim, W. Y. Yang, and C. G. Kang, *MIMO-OFDM wireless communications with MATLAB*. John Wiley & Sons, 2010.

[35] ``Ieee standard for local and metropolitan area networks part 16: Air interface for fixed broadband wireless access systems," *IEEE Std 802.16-2004 (Revision of IEEE Std 802.16-2001)*, pp. 1--857, 2004.

[36] A. Ahmedsaid, A. Amira, and A. Bouridane, ``Accelerating music method on reconfigurable hardware for source localisation," in *Circuits and Systems, 2004. ISCAS'04. Proceedings of the 2004 International Symposium on*, vol. 3, pp. III--369, IEEE, 2004.

[37] J. Allen, H. M. Davey, D. Broadhurst, J. K. Heald, J. J. Rowland, S. G. Oliver, and D. B. Kell, ``High-throughput classification of yeast mutants for functional genomics using metabolic footprinting," *Nature biotechnology*, vol. 21, no. 6, pp. 692--696, 2003.

[38] P. Liu, H. Zhang, and K. B. Eom, ``Active deep learning for classification of hyperspectral images," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 10, no. 2, pp. 712--724, 2017.

[39] P. J. Smith, M. Shafi, and L. M. Garth, ``Performance analysis for adaptive mimo svd transmission in a cellular system," in *2006 Australian Communications Theory Workshop*, pp. 49--54, Feb 2006.

[40] O. Bryt and M. Elad, ``Compression of facial images using the k-svd algorithm," *Journal of Visual Communication and Image Representation*, vol. 19, no. 4, pp. 270--282, 2008.

[41] D. Zhang, S. Chen, and Z.-H. Zhou, ``A new face recognition method based on svd perturbation for single example image per person," *Applied Mathematics and computation*, vol. 163, no. 2, pp. 895--907, 2005.

[42] J. R. Cavallaro and F. T. Luk, ``Cordic arithmetic for an svd processor," *Journal of parallel and distributed computing*, vol. 5, no. 3, pp. 271--290, 1988.

[43] R. Andraka, ``A survey of cordic algorithms for fpga based computers," in *Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays*, pp. 191--200, ACM, 1998.

[44] H. Jeong, J. Kim, and W. kyung Cho, ``Low-power multiplierless dct architecture using image correlation," *IEEE Transactions on Consumer Electronics*, vol. 50, pp. 262--267, Feb 2004.

[45] Y. H. Hu, ``Cordic-based vlsi architectures for digital signal processing," *IEEE Signal Processing Magazine*, vol. 9, pp. 16--35, July 1992.

[46] J. Valls, T. Sansaloni, A. Perez-Pascual, V. Torres, and V. Almenar, ``The use of cordic in software defined radios: a tutorial," *IEEE Communications Magazine*, vol. 44, pp. 46--50, Sept 2006.

[47] E. Antelo, J. Villalba, J. D. Bruguera, and E. L. Zapata, ``High performance rotation architectures based on the radix-4 cordic algorithm," *IEEE Transactions on Computers*, vol. 46, no. 8, pp. 855--870, 1997.

[48] Y. H. Hu and S. Naganathan, ``An angle recoding method for cordic algorithm implementation," *IEEE Transactions on Computers*, vol. 42, no. 1, pp. 99--102, 1993.

[49] M. Kuhlmann and K. K. Parhi, ``P-cordic: A precomputation based rotation cordic algorithm," *EURASIP Journal on Advances in Signal Processing*, vol. 2002, no. 9, p. 137251, 2002.

[50] M. D. Ercegovac and T. Lang, ``Redundant and on-line cordic: Application to matrix triangularization and svd," *IEEE Transactions on Computers*, vol. 39, no. 6, pp. 725--740, 1990.

[51] T.-B. Juang, S.-F. Hsiao, and M.-Y. Tsai, ``Para-cordic: Parallel cordic rotation algorithm," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 51, no. 8, pp. 1515--1524, 2004.

[52] H. Dawid and H. Meyr, ``The differential cordic algorithm: Constant scale factor redundant implementation without correcting iterations," *IEEE Transactions on Computers*, vol. 45, no. 3, pp. 307--318, 1996.

[53] P. K. Meher, J. Valls, T. B. Juang, K. Sridharan, and K. Maharatna, ``50 years of cordic: Algorithms, architectures, and applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 56, pp. 1893--1907, Sept 2009.

[54] J.-M. Delosme, ``Bit-level systolic algorithms for real symmetric and hermitian eigenvalue problems," *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 4, no. 1, pp. 69--88, 1992.

[55] J.-M. Delosme, ``Cordic algorithms: theory and extensions," in *33rd Annual Techincal Symposium*, pp. 131--145, International Society for Optics and Photonics, 1989.

[56] C.-Z. Zhan, Y.-L. Chen, and A.-Y. Wu, ``Iterative superlinear-convergence svd beamforming algorithm and vlsi architecture for mimo-ofdm systems," *IEEE Transactions on Signal Processing*, vol. 60, no. 6, pp. 3264--3277, 2012.

[57] D. Guenther, R. Leupers, and G. Ascheid, ``A scalable, multimode svd precoding asic based on the cyclic jacobi method," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 8, pp. 1283--1294, 2016.

[58] C.-H. Yang, C.-W. Chou, C.-S. Hsu, and C.-E. Chen, ``A systolic array based gtd processor with a parallel algorithm," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, no. 4, pp. 1099--1108, 2015.

[59] T. Kaji, S. Yoshizawa, and Y. Miyanaga, ``Development of an asip-based singular value decomposition processor in svd-mimo systems," in *Intelligent Signal Process-*

ing and Communications Systems (ISPACS), 2011 International Symposium on, pp. 1--5, IEEE, 2011.

[60] Y.-T. Hwang, K.-T. Chen, and C.-K. Wu, ``A high throughput unified svd/qrd precoder design for mimo ofdm systems," in *Digital Signal Processing (DSP), 2015 IEEE International Conference on*, pp. 1148--1151, IEEE, 2015.

[61] Y.-L. Chen, C.-Z. Zhan, T.-J. Jheng, and A.-Y. Wu, ``Reconfigurable adaptive singular value decomposition engine design for high-throughput mimo-ofdm systems," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 4, pp. 747--760, 2013.

[62] C. I. Ionita, B. Nazer, C. Feng, and B. Aazhang, ``An experimental study on the robustness of integer-forcing linear receivers," in *2017 IEEE International Conference on Communications (ICC)*, pp. 1--7, May 2017.

[63] A. Sakzad, J. Harshan, and E. Viterbo, ``Integer-forcing mimo linear receivers based on lattice reduction," *IEEE transactions on wireless communications*, vol. 12, no. 10, pp. 4905--4915, 2013.

[64] L. Schmalen, S. t. Brink, and A. Leven, ``Advances in detection and error correction for coherent optical communications: Regular, irregular, and spatially coupled ldpc code designs," *arXiv preprint arXiv:1704.04618*, 2017.

[65] M. Ahmadi and S. M. Fakhraie, ``Optimization of viterbi decoder parameters for wran system," in *2010 2nd International Conference on Future Computer and Communi-*

cation, vol. 3, pp. V3--53--V3--56, May 2010.

[66] WiMAX Forum, *Mobile Radio Conformance Tests*, test procedures ed., sep 2010.

[67] G. Golub and W. Kahan, ``Calculating the singular values and pseudo-inverse of a matrix," *Journal of the Society for Industrial and Applied Mathematics, Series B: Numerical Analysis*, vol. 2, no. 2, pp. 205--224, 1965.

[68] E. Rohani, G. Choi, and M. Lu, ``The normalized singular value decomposition of non-symmetric matrices using givens fast rotations," *arXiv preprint arXiv:1707.05189*, 2017.

[69] T. F. Chan and P. C. Hansen, ``Some applications of the rank revealing qr factorization," *SIAM Journal on Scientific and Statistical Computing*, vol. 13, no. 3, pp. 727--741, 1992.

[70] Y. Z. Huang, Y. H. Hsieh, and C. H. Lin, ``A flexible k-best sphere decoding kernel for configurable antennas and constellations," in *2017 IEEE International Conference on Consumer Electronics - Taiwan (ICCE-TW)*, pp. 115--116, June 2017.

[71] Z. Cai, Y. H. Wang, and S. Chattong, ``A low complexity and high throughput mimo detection vlsi design for mimo-ofdm systems," in *2016 IEEE 83rd Vehicular Technology Conference (VTC Spring)*, pp. 1--5, May 2016.

[72] D. Auras, R. Leupers, and G. Ascheid, ``Efficient vlsi architectures for matrix inversion in soft-input soft-output mmse mimo detectors," in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1018--1021, June 2014.

[73] C. Studer and H. Bolcskei, ``Soft-input soft-output single tree-search sphere decoding," *IEEE Transactions on Information Theory*, vol. 56, pp. 4827--4842, Oct 2010.

[74] Z. Guo and P. Nilsson, ``Algorithm and implementation of the k-best sphere decoding for mimo detection," *IEEE Journal on Selected Areas in Communications*, vol. 24, pp. 491--503, March 2006.

[75] S. Eberli, D. Cescato, and W. Fichtner, ``Divide-and-conquer matrix inversion for linear mmse detection in sdr mimo receivers," in *2008 NORCHIP*, pp. 162--167, Nov 2008.

[76] D. Garrett, L. Davis, S. ten Brink, B. Hochwald, and G. Knagge, ``Silicon complexity for maximum likelihood mimo detection using spherical decoding," *IEEE Journal of Solid-State Circuits*, vol. 39, pp. 1544--1552, Sept 2004.

[77] R. Shariat-Yazdi and T. Kwasniewski, ``Configurable k-best mimo detector architecture," in *2008 3rd International Symposium on Communications, Control and Signal Processing*, pp. 1565--1569, March 2008.

[78] M. Rahman, E. Rohani, J. Xu, and G. Choi, ``An improved soft decision based mimo detection using lattice reduction," *International Journal of Computer and Communication Engineering*, Apr 2014.

[79] R. Schwarz, A. Knopp, D. Ogermann, C. Hofmann, and B. Lankl, ``Optimum-capacity mimo satellite link for fixed and mobile services," pp. 209--216, 2008.

[80] A. Knopp, R. T. Schwarz, D. Ogermann, C. A. Hofmann, and B. Lankl, ``Satellite system design examples for maximum mimo spectral efficiency in los channels," pp. 1--6, 2008.

[81] R. Schwarz, A. Knopp, and B. Lankl, ``The channel capacity of mimo satellite links in a fading environment: A probabilistic analysis," pp. 78--82, 2009.

[82] B. Noureddine and I. Leyla, ``Design of mimo satellite system: Inter-antenna spacing determination and possible enhancement of capacity," pp. 351--364, 2011.

[83] A. Vanelli-Coralli, G. Corazza, G. Karagiannidis, P. Mathiopoulos, D. Michalopoulos, C. Mosquera, S. Papaharalabos, and S. Scalise, ``Satellite communications: Research trends and open issues," pp. 71--75, 2007.

[84] H.-Y. Shen and S. Kalyanaraman, ``Asynchronous cooperative mimo communication," pp. 1--9, 2007.

[85] S. Cui, A. J. Goldsmith, and A. Bahai, ``Energy-efficiency of mimo and cooperative mimo techniques in sensor networks," *Selected Areas in Comm, IEEE Journal on*, vol. 22, no. 6, pp. 1089--1098, 2004.

[86] L. Cristofoli, A. Henglez, J. Benfica, L. Bolzani, F. Vargas, A. Atienza, and F. Silva, ``On the comparison of synchronous versus asynchronous circuits under the scope of conducted power-supply noise," pp. 1047--1050, 2010.

[87] R. D. Jorgenson, L. Sorensen, D. Leet, M. S. Hagedorn, D. R. Lamb, T. H. Fridell, and W. P. Snapp, ``Ultralow-power operation in subthreshold regimes applying

clockless logic," *Proceedings of the IEEE*, vol. 98, no. 2, pp. 299--314, 2010.

[88] K.-S. Chong, K.-L. Chang, B.-H. Gwee, and J. S. Chang, ``Synchronous-logic and globally-asynchronous-locally-synchronous (gals) acoustic digital signal processors," *Solid-State Circuits, IEEE Journal of*, vol. 47, no. 3, pp. 769--780, 2012.

[89] P. Bhagawat, R. Dash, and G. Choi, ``Dynamically reconfigurable soft output mimo detector," in *Computer Design, 2008. ICCD 2008. IEEE International Conference on*, pp. 68--73, Oct 2008.

[90] K. Gunnam, G. Choi, W. Wang, and M. Yeary, ``Multi-rate layered decoder architecture for block ldpc codes of the ieee 802.11 n wireless standard," pp. 1645--1648, 2007.

[91] T. Lin, K.-S. Chong, J. Chang, and B.-H. Gwee, ``An ultra-low power asynchronous-logic in-situ self-adaptive v_{dd} system for wireless sensor networks," *Solid-State Circuits, IEEE Journal of*, vol. 48, pp. 573--586, Feb 2013.

[92] Q. Zhou and K. Mohanram, ``Cost-effective radiation hardening technique for combinational logic," in *Computer Aided Design, 2004. ICCAD-2004. IEEE/ACM International Conference on*, pp. 100--106, Nov 2004.